ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

РАЗРАБОТКА WEB-СЕРВИСА ДЛЯ ЗАЩИЩЕННОГО ONLINE-ОФОРМЛЕНИЯ ДОГОВОРОВ АРЕНДЫ

Выпускная квалификационная работа
обучающегося по направлению подготовки
09.03.02 Информационные системы и технологии
очной формы обучения, группы 12001508
Рыженко Наталии Олеговны

Научный руководитель

Ст.преподаватель
Смышляев А.Г.

БЕЛГОРОД 2019

РЕФЕРАТ

Разработка web-сервиса для защищенного online-оформления договоров аренды – Рыженко Наталия Олеговна, выпускная квалификационная работа бакалавра Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 63, включая приложения 79, количество рисунков 64, количество таблиц 6, количество использованных источников 32.

КЛЮЧЕВЫЕ СЛОВА: web-сервис, online-оформление договора, аренда вещей, квалифицированная/неквалифицированная подпись.

ОБЪЕКТ ИССЛЕДОВАНИЯ: аренда товаров в сети Интернет.

ПРЕДМЕТ ИССЛЕДОВАНИЯ: достоинства и недостатки предоставления товаров в аренду в сети Интернет.

ЦЕЛЬ РАБОТЫ: предоставление клиентам средства, позволяющего быстро и безопасно заключать сделки по аренде товаров, в том числе, за счет развитого функционала работы с личным кабинетом арендатора/арендодателя, применения средств электронной подписи при заключении договоров, отслеживания статуса перевода денежных средств между участниками сделки.

ЗАДАЧИ ИССЛЕДОВАНИЯ: исследовать заключение договора аренды по российскому гражданскому праву; рассмотреть способы шифрования информации, технологию создания и применения цифровой подписи; учесть федеральный законом "Об электронной подписи"; изучить существующие сервисы по предоставлению услуг аренды в сети; рассмотреть существующие вспомогательные программы/библиотеки для создания сертификатов и цифровых подписей.

МЕТОДЫ ИССЛЕДОВАНИЯ: экспериментальный метод, сравнение с существующими данными.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: в результате работы был разработан web-сервис для защищенного online-оформления договоров аренды.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современной жизни все чаще поднимается тема актуальности информационных систем и технологий. Внедрение новых компьютерных технологий поглотило все сферы жизни. Рутинная работа все чаще заменятся автоматизированной, на новый уровень выходит компьютерный интеллект, появляются новые сервисы для предоставления услуг.

Благодаря доступности компьютерной техники и сети Интернет большинство людей активно используют веб-сервисы для передачи информации, записи на прием, совершения онлайн покупки и т.д. Данная тенденция привела к развитию каналов связи, усовершенствованию алгоритмов шифрования и обеспечению защищенного обмена данными. Далее пришла необходимость в заключении гражданско-правовых сделок, в оказании государственных и муниципальных услуг в сети Интернет. В связи с этим возникли новые законы для урегулирования таких ситуаций и приданию таким сделкам юридическую силу.

Развитие популярности shared economy – так называемой экономики совместного потребления, когда потребители предпочитают не владеть, а пользоваться вещами, привело к появлению сервисов не только по аренде квартир, машин, но также по аренде самых обычных бытовых вещей вплоть до посуды. Совместное потребление основано на идее, что удобнее платить за временный доступ к продукту, чем владеть им постоянно [1]. А также созданы все условия для заключения договора аренды прямо из дома.

К сожалению, практику аренды вещей в России едва ли можно назвать популярной. Все мы привыкли к тому, что даже за самой незначительной или временной необходимостью нужно идти в магазин и приобретать, оплачивая ее полную стоимость. Хотя нередко наступают такие случаи, когда вещь нужна на короткий срок: большой чемодан – для поездки в Лондон, лодка – для отдыха с друзьями. И возникает необходимость в приобретении этого

товара. Но стоит ли покупать вещь на один раз, чтобы она потом пылилась и занимала место в небольшой квартире.

Новые возможности сети интернет позволяют создать веб-сервис, который будет удобен в использовании, позволит быстро и без рисков взять в аренду вещь для временного использования – будь то предметы декора, рабочие инструменты или детские игрушки. То же самое касается владельцев – не расставаясь со своей вещью навсегда, поможет сдать в аренду тем, кто в ней нуждается.

Целью выпускной квалификационной работы: предоставление клиентам средства, позволяющего быстро и безопасно заключать сделки по аренде товаров, в том числе, за счет развитого функционала работы с личным кабинетом арендатора/арендодателя, применения средств электронной подписи при заключении договоров, отслеживания статуса перевода денежных средств между участниками сделки.

Были поставлены следующие задачи:

– исследовать заключение договора аренды по российскому гражданскому праву;

– рассмотреть способы шифрования информации, технологию создания и применения цифровой подписи;

– учесть федеральный закон "Об электронной подписи";

– изучить существующие сервисы по предоставлению услуг аренды в сети;

– рассмотреть существующие вспомогательные программы/библиотеки для создания сертификатов и цифровых подписей;

– разработать web-сервис для предоставления услуги защищенного онлайн-оформления договора аренды.

Пояснительная записка выполнена на 63 страницах без приложения, содержит 64 рисунков, 6 таблиц.

1 Аналитическая часть

1.1 Общая характеристика web-сервиса по предоставлению товара в аренду

Объектом исследования является аренда товаров в сети Интернет.

Предметом исследования является обеспечение удобства и безопасности аренды товара через Интернет.

Магазины по аренде товара всегда имели ряд преимуществ перед классическими магазинами. Нет необходимости беспокоиться об остатках, как ориентироваться на сезонность. Экономия на издержках позволяет на этапах открытия бизнеса быстро выйти на точку окупаемости и начать работать в плюс [2]. В таблице 1 представлена экономика бизнеса по аренде одежды.

Таблица 1 – Экономика бизнеса по сдаче одежды в аренду

Ком-пания	Год запуска	Стар-товые вложения, тыс. руб.	Средний чек одного цикла аренды, тыс. руб.	Количество циклов аренды, которые переживает вещь	Выручка, тыс. руб./мес.	Чистая прибыль в высокий сезон, тыс. руб./мес.	Рентабельность, %
Гардероб	2015	400	4,9–7,5	4–20	800	360	45
Moussa Project	2009	3000	12	4–5	500–2000	90–360	18
Rentmania (только сегмент одежды)	2012	200	4	5–10	500	90	15
Dress Up Bar	2014	3000	4,5	5–30	100–2200	100–600	13

Как видно из таблицы 1, прокат вещей является успешным бизнесом. Один из недостатков таких сервисов это обязательность наличия собственного помещения, в котором будут предоставлять товары в аренду, заключаться договора.

Еще один способ предоставлять товары в аренду – создать web-сервис, в котором каждый желающий может разместить свой товар, указать параметры аренды: стоимость, срок и залог аренды – и предоставлять их в аренду.

С 2011 года появилась возможность подписывать договора в сети Интернет при помощи электронной подписи, это гласит Федеральный закон «Об электронной подписи» от 06.04.2011 N 63-ФЗ. Выписка из 6 статьи: «Информация в электронной форме, подписанная квалифицированной электронной подписью, признается электронным документом, равнозначным документу на бумажном носителе, подписанному собственноручной подписью, и может применяться в любых правоотношениях в соответствии с законодательством Российской Федерации…» [3].

Данное нововведение позволит арендаторам/арендодателям быстро по одному клику взять/сдать товар в аренду и заключить электронный договор, который будет иметь юридическую силу.

Задачами такого веб-сервиса являются:

– предоставить владельцам товаров некую платформу для размещения своих предложений;

– предоставить удобный способ просмотра вещей, выставленных на аренду, с помощью фильтрации, сортировок, поисков;

– организовать взаимодействие арендатора и арендодателя при возникновении желания взять вещь в аренду;

– создать договор аренды, имеющий юридическую силу;

– настроить работу сервиса, предусматривающего защиту от мошенничества;

– получать прибыль от состоявшихся сделок.

1.2 Юридически значимый online-договор аренды

Чтобы владельцу защитить свое имущество от пропажи или ущерба, а арендатору иметь право пользоваться арендованной вещью до окончания

установленного срока, заключается договор аренды. Договор аренды – соглашение, в силу которого одна сторона – арендодатель, обязуется предоставить другой стороне – арендатору, имущество за плату во временное владение и пользование или во временное пользование, а арендатор обязуется уплачивать арендную плату [4].

Данный документ позволяет в случае невыполнения одного из установленных правил любой стороне обратиться за поддержкой в суд для урегулирования возникшей ситуации.

Договор должен включать следующие пункты:

– дату создания договора;

– данные об арендодателе/арендаторе;

– данные об объекте аренды, которые позволяют определенно установить передаваемое в аренду имущество;

– срок аренды и арендная плата;

– права и обязанности арендодателя/арендатора;

– подпись двух сторон.

Одним немаловажным пунктом заключения договора является подпись. Как уже упоминалось, с 2011 года в качестве личной подписи для заключения договоров в Интернете может служить электронная подпись (ЭП).

ЭП – информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию [5].

Существует два вида ЭП: усиленная электронная подпись и простая. Усиленная в свою очередь делится на неквалифицированную и квалифицированную электронную подпись.

Простая ЭП – это логин/пароль или код из СМС, которые вводятся для авторизации в интернет-магазине, портале «Госуслуги» или внутренней корпоративной сети, подтверждая свою личность.

Неквалифицированная ЭП – это зашифрованная комбинация символов, которая подтверждает личность пользователя и позволяет обнаружить внесение изменений в документ после его подписания [6].

Квалифицированная подпись обладает теми же возможностями, что и неквалифицированная, отличием является то, что она создается с использованием средств шифрования, сертифицированных ФСБ.

В таблице 2 представлены свойства электронных подписей.

Таблица 2 – Свойства электронных подписей

	Простая	Неквалифицированная	Квалифицированная
Способ получения	При регистрации на сайте	В любом УЦ	В аккредитованном УЦ
Защита подписанного документа	Не защищает документ от подделки	Защищает документ от подделки	Защищает документ от подделки
Юридическая значимость	Требует соглашения о признании	Требует соглашения о признании	Равна собственноручной подписи
Где хранятся	На любом носителе	На любом носителе	Защищенный носитель (Рутокен, eToken)
Стоимость	Бесплатно	Бесплатно или от 200 рублей	От 500 рублей

Исходя из свойств представленных видов подписей были выбраны квалифицированная и неквалифицированная подпись в качестве ЭП при подписании договора.

1.3Анализ существующих аналогов программных средств

При поиске существующих аналогов был сделан вывод, что online-аренда вещей в сети Интернет встречается только в крупных городах, таких

как Москва, Санкт-Петербург, Ростов. К тому же, по сравнению с количеством интернет магазинов, существует одна или пару таких служб на весь город.

Рассмотрим некоторые примеры аналогов, представленные в таблице 3.

Таблица 3 – Анализ существующих веб-сервисов по предоставлению аренды

Название	Удобство	Возможность добавить свой товар для аренды	Залог	Разнородность предоставляемых в аренду вещей	Самовывоз	Заключение online-договора
Beriigray (Белгород)	+	-	-	- (только детские игрушки)	-	-
Toyrent (Ростов)	+	-	-	- (только детские игрушки)	-	-
Arendorium (Москва, Санкт-Петербург)	+	+	+ (полная стоимость)	+	-	-
Rentmania (Москва, Санкт-Петербург)	+	+	+ (полная стоимость товара)	+	+	+ (при помощи простой ЭП)

В таблице выше представлены самые популярные сервисы по аренде товаров. Удобство использования этих web-сервисов заключается в просмотре всех имеющихся товаров для аренды прямо из дома. Многие платформы ориентированы на предоставление в аренду только своего товара, что делает невозможным всем желающим разместить свой товар на сайте. При выборе определенного товара арендатор обязан заплатить арендную плату и залог, если это определено условиями данного сервиса. Не все сервисы могут предложить разнородные товары – от детской игрушки до аренды автомобиля. Также достоинством сервиса является наличие самовывоза, т.к. не каждый арендатор предпочитает тратить деньги на доставку курьером, предлагаемым сервисом. К тому же большинство web-сервисов не имеют возможность

заключать online-договора аренды, единственным существующим web-сервисом с такой возможностью является Rentmania, но и она предоставляет заключение договора только при помощи простой ЭП, что не дает полную защиту от мошенничества.

Проанализировав все преимущества и недостатки различных web-сервисов по предоставлению товаров в аренду, было сделано решение создать свое приложение, которое бы обладала преимуществами, представленными в таблице 4.

Таблица 4 – Характеристика нового web-сервиса

Название	Удобство	Возможность добавить свой товар для аренды	Залог	Разнородность предоставляемых в аренду вещей	Самовывоз	Заключение online-договора
RentZone Белгород	+	+	предусматривается арендодателем	+	+	+ (при помощи усиленной ЭП)

Гарантом того, что арендуемая вещь будет возвращена целой и в назначенный срок служит договор аренды.

Вывод по первому разделу

В данном разделе было приведено экономическое обоснование создания web-сервиса по предоставлению товара в аренду, его основные возможности и выбран способ подписание электронного договора аренды, также были рассмотрены существующие аналоги web-сервисов по предоставлению товаров в аренду, кроме того продемонстрированы их достоинства и недостатки, в связи с чем было принято решение о целесообразности создания нового web-сервиса, который обладал бы всеми вышеперечисленными достоинствами.

2 Моделирование АИС

2.1 Разработка функциональной модели

Создание современных информационных систем представляет собой сложную задачу. Прежде чем перейти к основным шагам разработки, необходимо понять и описать бизнес-логику предметной области.

Перед разработкой ИС должны быть четко сформулированы основные возможности разрабатываемой системы: какие функции будут заложены в систему и как будут взаимодействовать между собой ее составляющие.

Сформулировать одно представление поможет составление функциональной модели, которая предназначена для описания процессов обработки данных в рамках выделенной предметной области с различных точек зрения [7].

В настоящее время двумя наиболее популярными методологиями построения функциональных моделей являются SADT и DFD.

При помощи данных методологий можно разбить исходное представление сложной системы на отдельные составные части, создавая серию иерархически взаимосвязанных диаграмм с сопутствующей документацией. В таком случае один из основных процессов представляется в виде более детальных процессов на диаграмме нижнего уровня за счет декомпозиции. Это приводит к тому, что диаграмма с общими функциями конкретизируется на ряд более детальных диаграмм [8].

Одним из самых популярных CASE-средств верхнего уровня является ERwin, который поддерживает вышеупомянутые методологии. С его помощью создаются функциональные модели, предназначенные не только для описания существующих бизнес-процессов в организации, а также для описания взаимосвязи потоков информации и процессов в программном обеспечении [9].

Функциональная модель web-сервиса по аренде товаров была построена с помощью предложенного CASE-средства ERwin (AllFusion Process Modeler).

Данная модель была написана с использование методологии IDEF0, IDEF3 и DFD. Построение начинается с описания первоначальной глобальной функции – контекстной диаграммы (диаграмма верхнего уровня), которая является вершиной древовидной структуры диаграмм и показывает назначение системы (основную функцию), и ее взаимодействие с внешней средой. Контекстная диаграмма web-сервиса по аренде товаров представлена на рисунке 1.

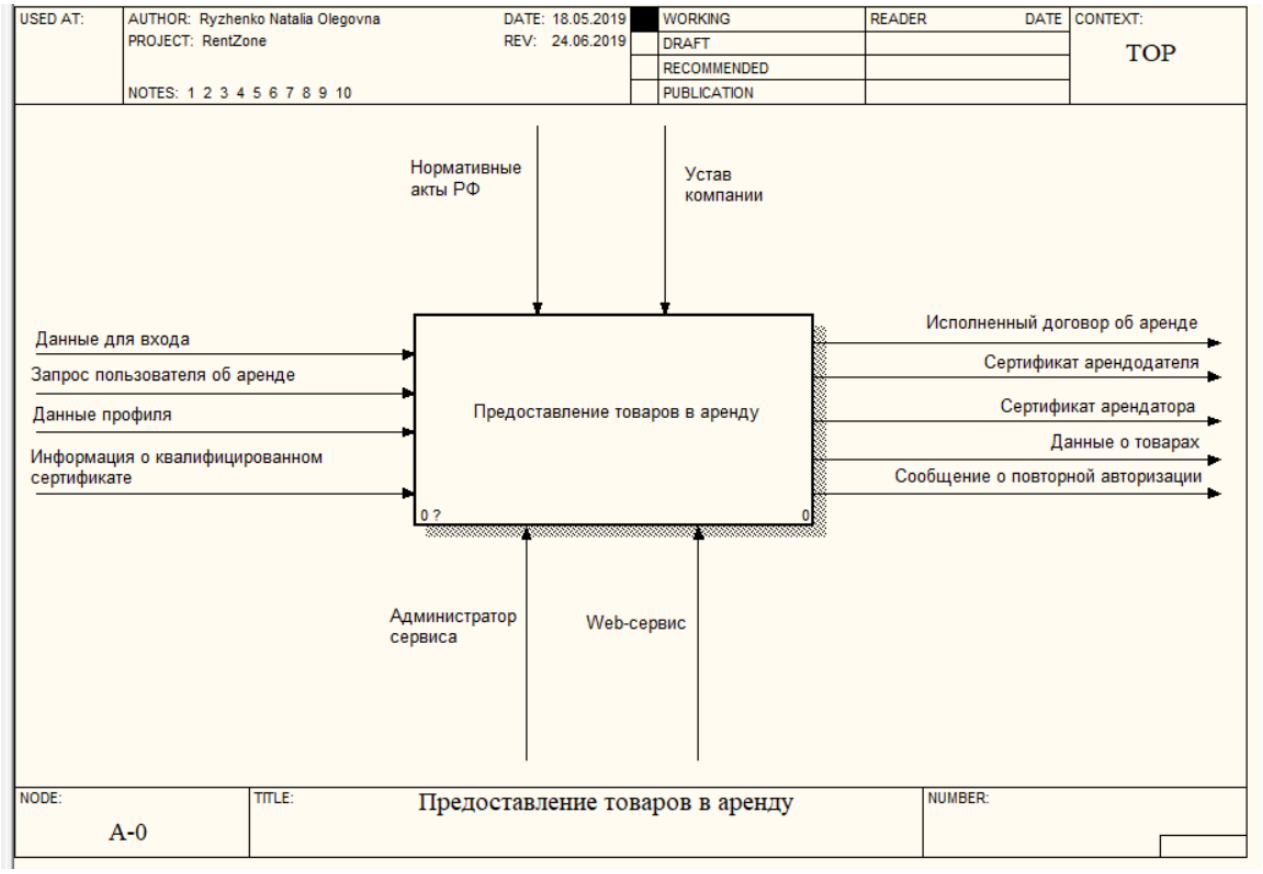

Рисунок 1 – Контекстная диаграмма

Контекстная диаграмма состоит из одного блока и стрелок механизма, управления, входных данных и выходных в соответствии с методологией IDEF0. Функционирование данного сервиса начинается с поступление входных данных – введенные пользователем данные для входа, профиля, запрос об аренде и информация о квалифицированном сертификате.

Механизм, применяемый в данной системе – администратор сервиса и сам web-сервис. На выходе образуются такие данные как исполненный договор об аренде, сертификаты арендатора/арендодателя и данные о товарах. Управляется система нормативными актами РФ и уставом компании.

После чего была проведена функциональная декомпозиция – система разбивается на подсистемы, и каждая подсистема описывает более мелкие функции. Затем каждая подсистема была разбита еще на более мелкие и так далее до достижения нужных нам процессов.

При построении диаграммы декомпозиции первого уровня исходная функция разбивается на четыре блока в соответствии с рисунком 2.

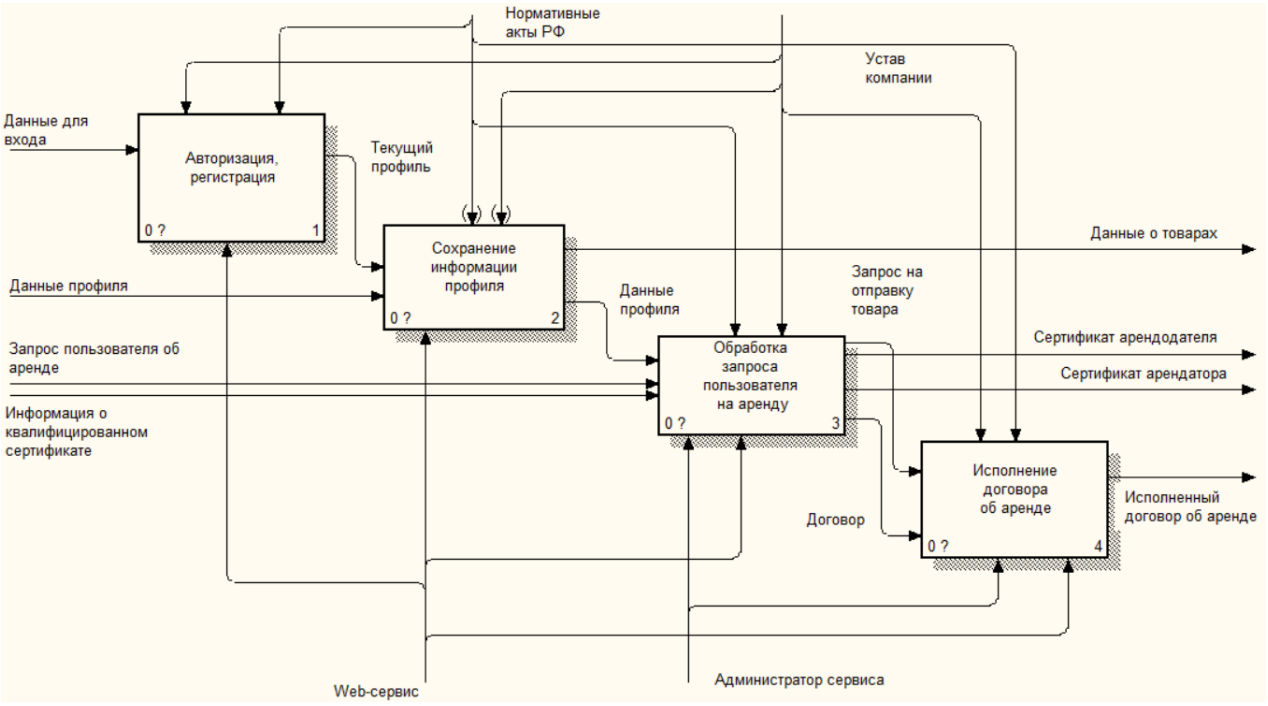

Рисунок 2 – Декомпозиция 1-го уровня

На рисунке 2 отображены основные процессы, которые необходимы для реализации аренды товаров в сети Интернет:

– авторизация/регистрация;

– сохранение информации профиля;

– обработка запроса пользователя на аренду;

– исполнение договора об аренде.

Данные для входа поступают в первый функциональный блок – «Авторизация/регистрация». Данный блок содержит в себе еще 4 функции, которые подробно описывают работу авторизации/регистрации и представлены на рисунке 3.

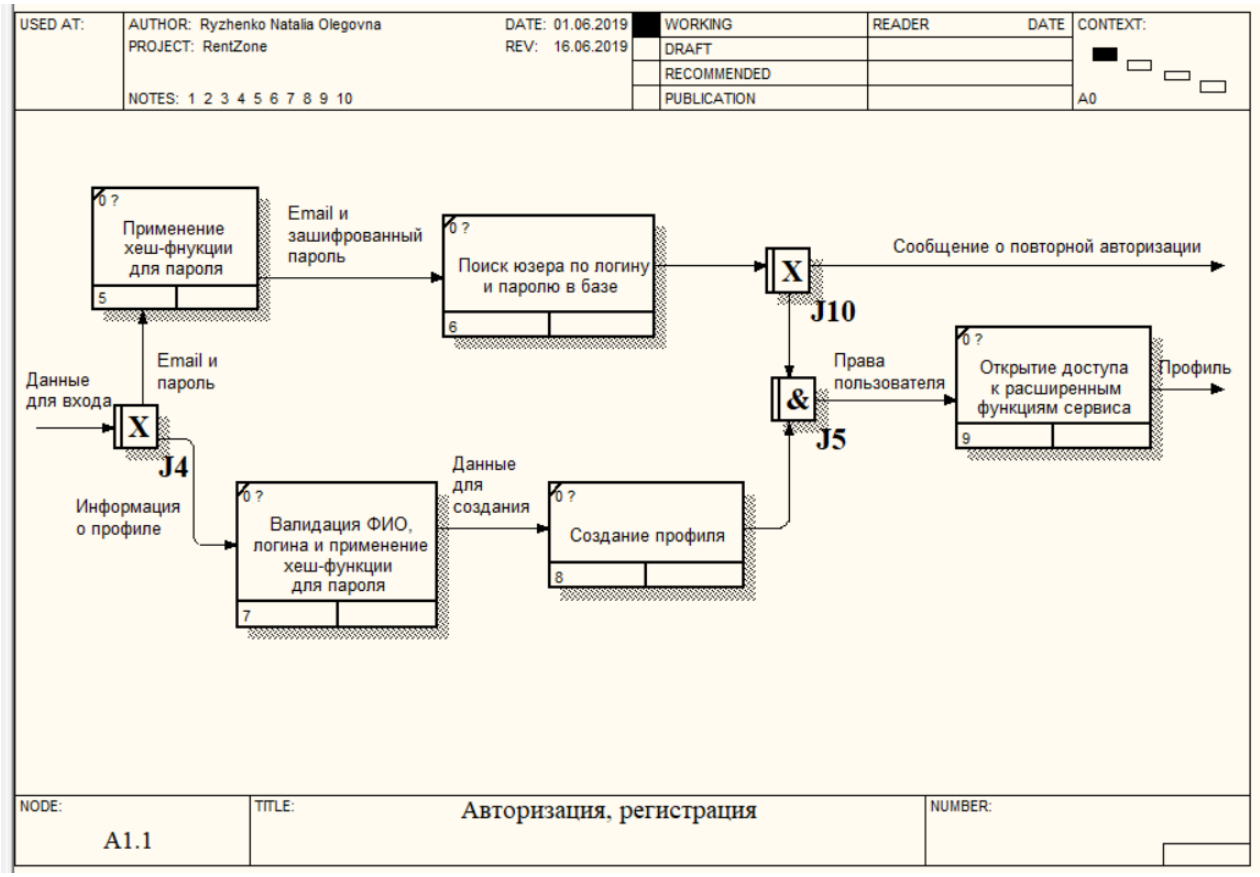

Рисунок 3 – Декомпозиция блока «Авторизация/регистрация»

Декомпозиция представляет основные действия при авторизации пользователя или регистрации. При авторизации пользователь вводит email и пароль и отправляет на сервис. Сервис шифрует пароль при помощи хеш-функции и находит совпадение email-а и пароля у пользователей из базы. Если совпадения не найдены, отправляет сообщение о неверном вводе данных для авторизации. При регистрации необходимые данные проходят валидацию, пароль шифруется и создается новая запись в базе пользователей. После успешной авторизации/регистрации пользователь может попасть на любую страничку в соответствии с правами.

После блока «Авторизация, регистрация» располагается блок по сохранению информации профиля, который содержит в себе 3 функции, представленные на рисунке 4.

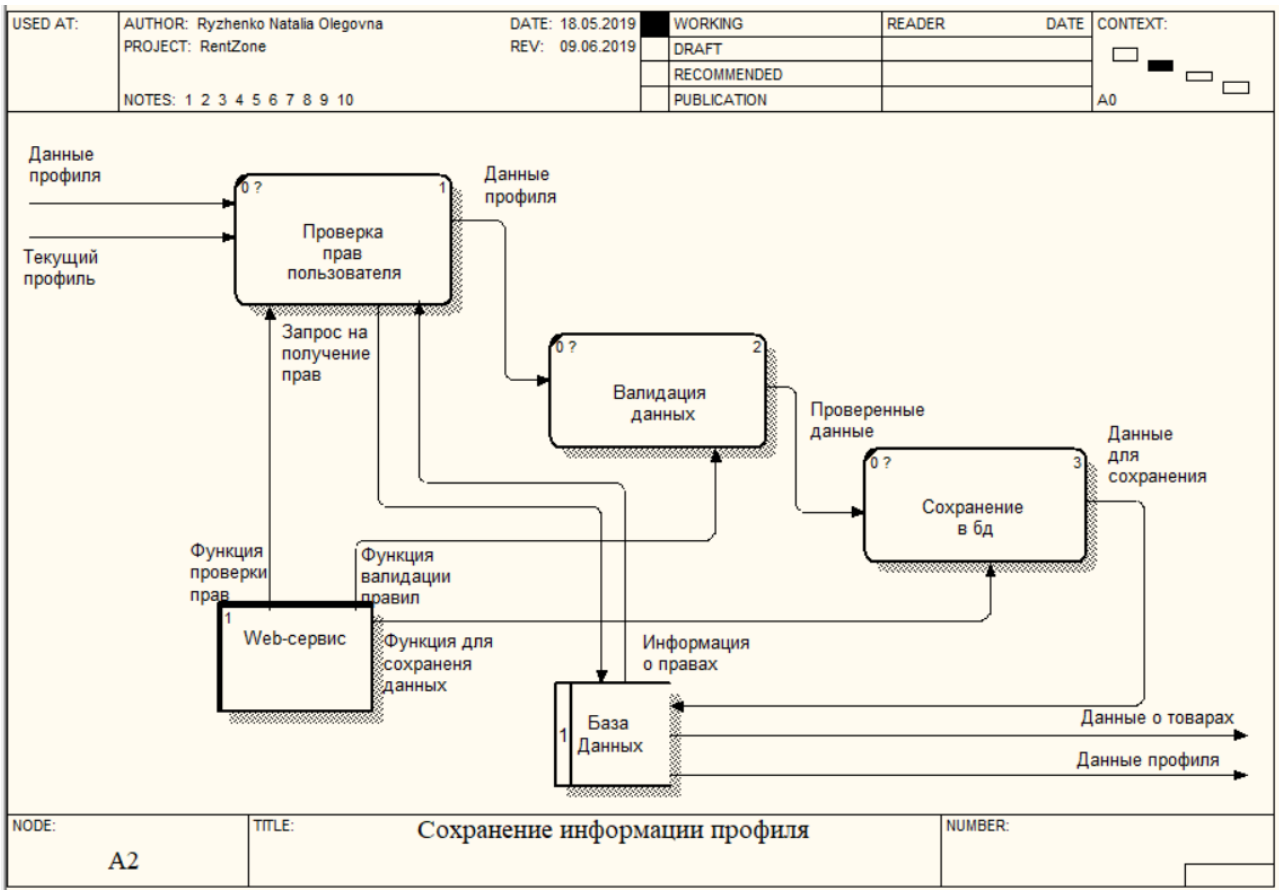

Рисунок 4 – Декомпозиция блока «Сохранение информации профиля»

На вход поступает текущий профиль и новые данные профиля. К новым данным профиля относятся личные данные пользователя, данные о товаре для предоставления в аренду. Далее с помощью реализованных функций проверяются права пользователя на создание или редактирования данного профиля, новые данные проходят валидацию после чего сохраняются в базе данных.

Для аренды товара пользователь указывает сроки аренды, остальные необходимые данные и создает запрос на аренду товара. На рисунке 5 представлена декомпозиция блока «Обработка запроса пользователя на аренду».

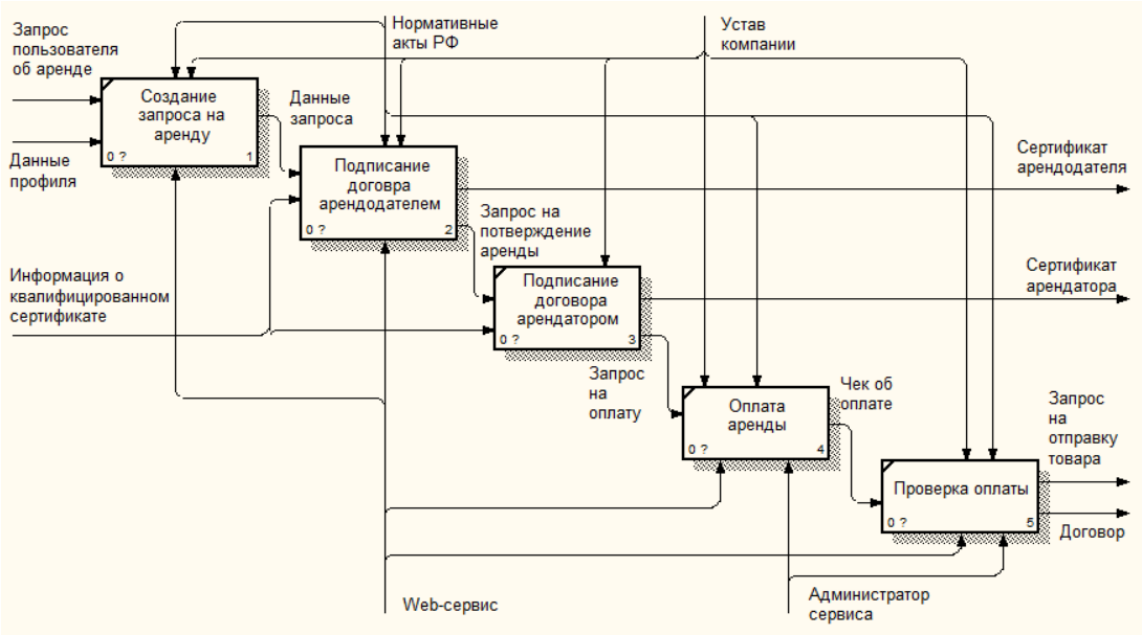

Рисунок 5 – Декомпозиция блока «Обработка запроса на аренду»

Отображаемые на рисунке блоке работают под управлением web-сервиса. Арендодатель в случае одобрения подтверждает запрос, тем самым подписывая договор. Договор создается и подписывается с учетом нормативных актов РФ и по уставу компании. Далее арендатору также остается подписать договор и предоставить чек об оплате. Оплата приходит на счет сервиса и администратор подтверждает оплату, как только сумма поступит на счет.

Подписание договора арендодателем представлено на рисунке 6.

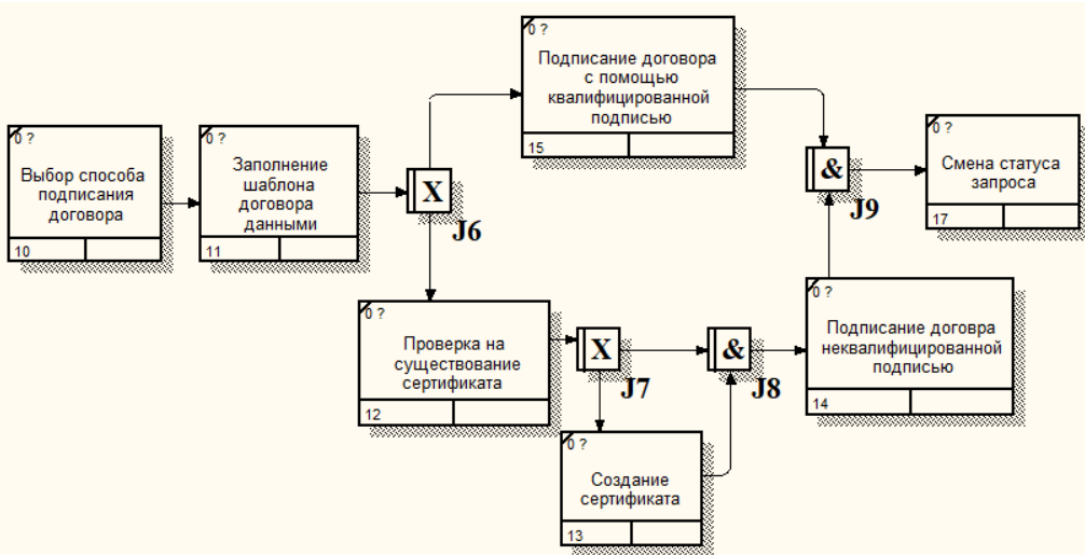

Рисунок 6 – Блок «Подписание договора арендодателем»

При утверждении запроса арендатора на аренду товара, владелец товара выбирает способ подписания договора: с помощью квалифицированной подписи или простой. Шаблон договора аренды заполняется данными запроса и подписывается выбранным способ. Далее статус запроса меняется на «Ожидание подтверждения запроса арендатором». Арендатор по такой же схеме подписывает свой договор.

Запрос на отправку товара и подписанные договора поступают в последний блок декомпозиции – «Исполнение договора об аренде». Работа этого блока отображена на рисунке 7.

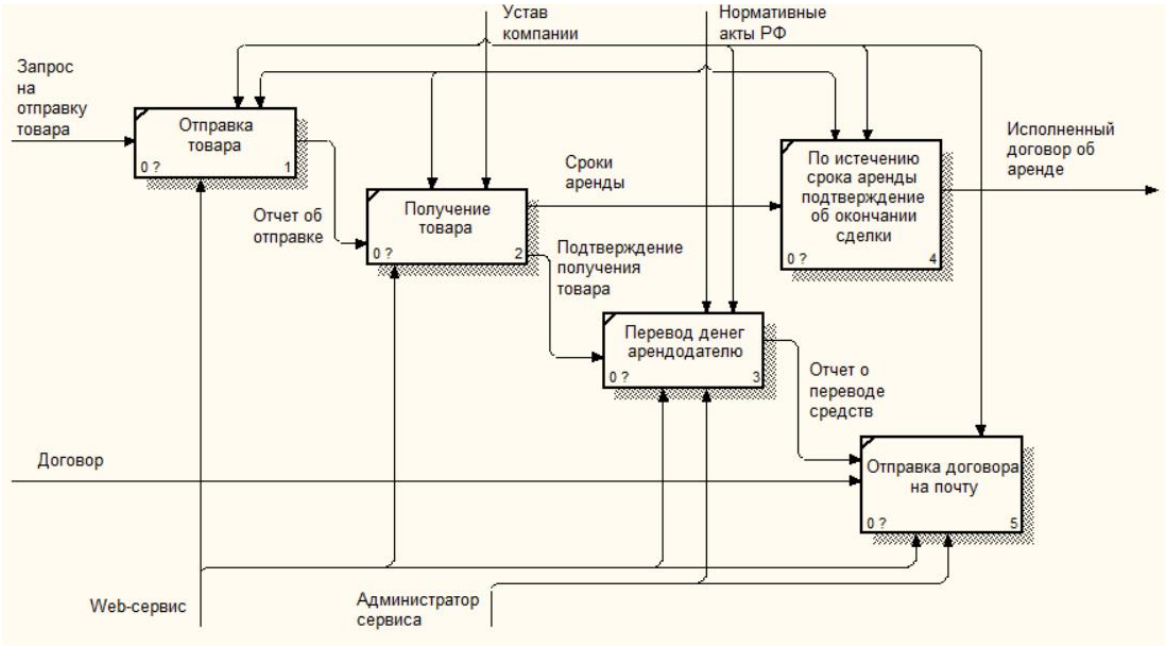

Рисунок 7 – Декомпозиция блока «Исполнение договора об аренде»

После заключения договора об аренде наступает его исполнение. Владелец товара должен отправить товар и загрузить фотографию или чек подтверждающий отправку товара арендатору. После подтверждения арендатором о получении товара, администрация пересылает арендодателю деньги со счета и всем сторонам отправляется договор на почту. По истечению срока аренды, арендатор должен отправить вещь обратно владельцу, и владелец подтверждает окончание сделки.

2.2 Разработка информационной модели

Информационная модель – совокупность информации, характеризующая существенные свойства и состояния объекта, процесса, явления, а также взаимосвязь с внешним миром [10].

Процесс построения информационной модели состоит из следующих шагов:

– определение сущностей;

– определение зависимостей между сущностями;

– задание первичных и альтернативных ключей;

– определение атрибутов сущностей;

– приведение модели к требуемому уровню нормальной формы;

– переход к физическому описанию модели (назначение соответствий имя сущности – имя таблицы, атрибут сущности – атрибут таблицы);

– задание триггеров, процедур и ограничений;

– генерация базы данных.

Модель базы данных была создана при помощи программного средства Navicat Data Modeler. Navicat Data Modeler является мощным средством для проектирования базы данных. Он позволяет визуально создавать логические, физические модели данных, сложные сценарии SQL/DDL и поможет преобразовать созданные модели со сложными взаимосвязями в SQL-скрипт простым щелчком мыши [11].

Navicat Data Modeler поддерживает различные СУБД – MySQL, MariaDB, Oracle, SQL Server, PostgreSQL и SQLite. Особенностью данного средства является обратное проектирование. Получить готовые модели можно из уже существующих баз данных, создав подключение к своей СУБД и выбрав название базы данных. Действия по импортированию готовой структуры отображены на рисунке 8.

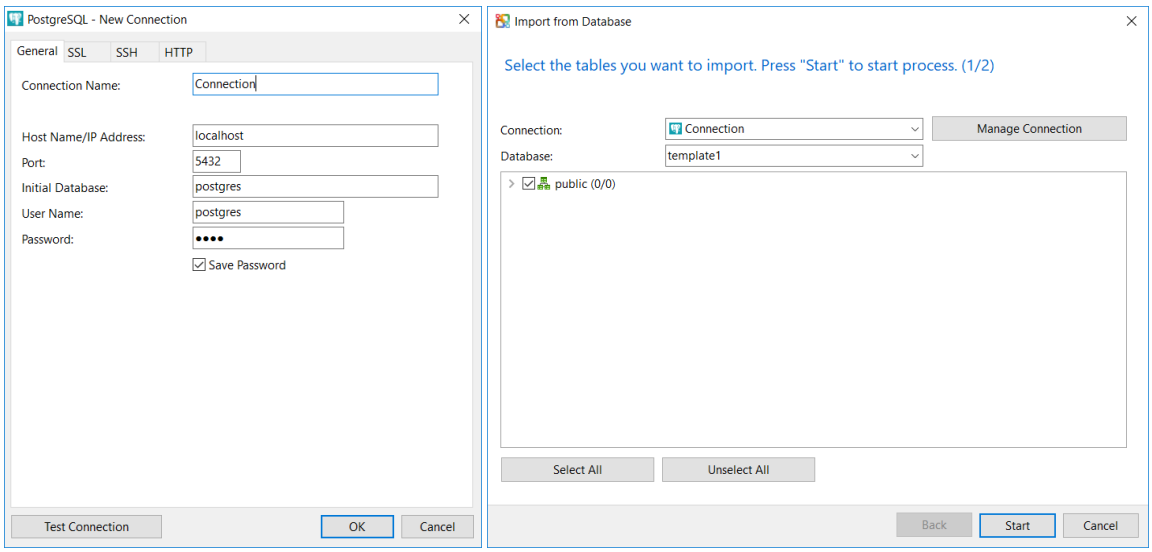

Рисунок 8 – Импортирование структуры из имеющейся базы данных

Это позволит представить готовую базу в удобном для просмотра виде, убедиться в правильности определения взаимосвязей различных элементов, таких как атрибуты, связи, индексы, ограничения уникальности, комментарии и другие объекты.

Самым сложным этапом при проектировании базы является нормализация. Нормализация – это процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости [12].

При избыточности данных непродуктивно используется свободное место на диске и затрудняется обработка или изменение данных [13]. К примеру, если потребуется изменить данные, которые используются в нескольких местах, то придется сделать изменения во всех этих местах. Несогласованные зависимости также негативно влияют на структуру базы. Несогласованные зависимости возникают тогда, когда данные указываются не в тех таблицах, то есть при поиске данных человек не сможет логически предположить нахождение этих данных ориентируясь только по названию таблиц, что займет значительное количество времени при поиске необходимых данных.

Чтобы избежать сложностей при дальнейшем использовании базы данных были придуманы 6 правил «нормальных форм». Как правило, для выполнения нормализации приходится хорошо проанализировать все данные, что иногда приводит к созданию дополнительных таблицы, а то и к полному изменению структуры.

На рисунке А.1 представлена уже готовая физическая модель web-сервиса по аренде товаров, которая была создана с помощью вышеописанного программного средства.

В данных таблицах не будут содержаться повторяющиеся данные в различных строках, а также ячейки не будут содержать больше одного значения, что соответствует «первой нормальной форме». Также все неключевые поля не зависят от других неключевых полей в табице, что соовествует «второй и третьей нормальным формам».

После создания физической модели можно переходить к созданию базы данных. Для этого мы воспользовались особенностью Navicat Data Modeler и сгенерировали SQL-скрипт. Действие по генерации скрипта представлено на рисунке 9.

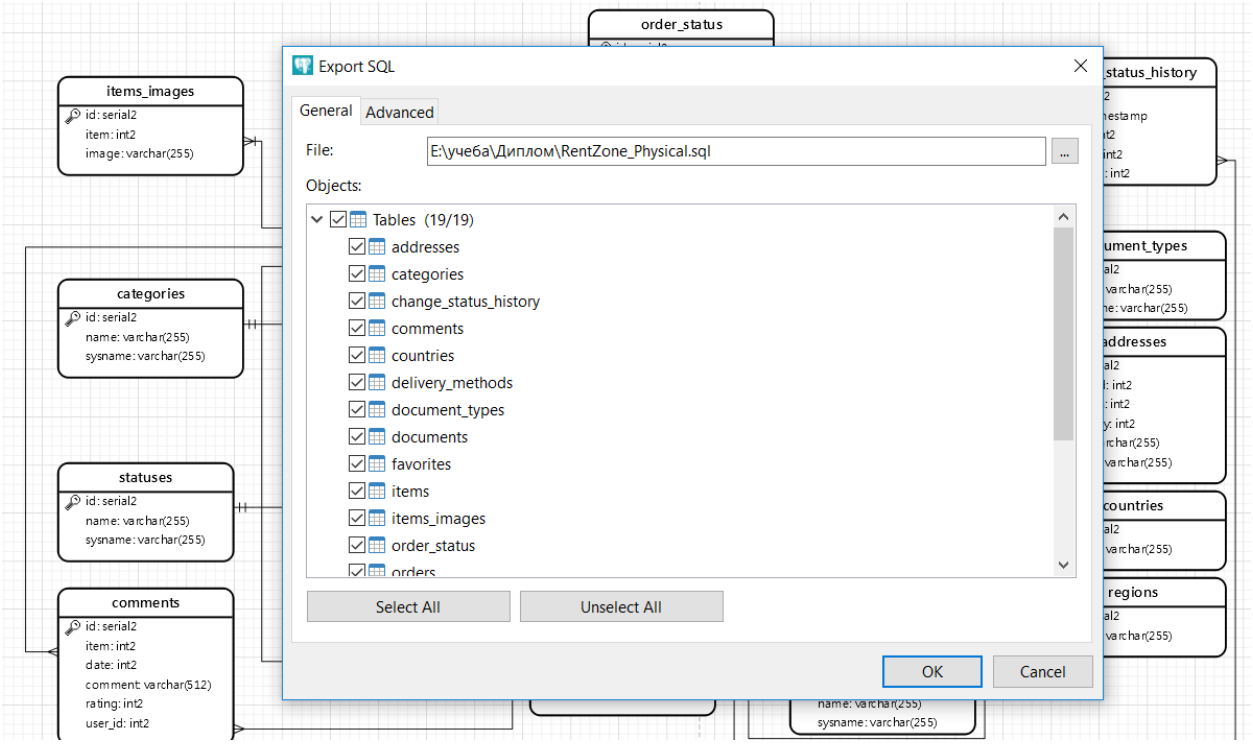

Рисунок 9 – Генерация SQL-скрипта

На вкладке «Advanced» выбираем СУБД PosrtgreSQL версии 9.3 и получаем скрипт, представленный на рисунке 10.

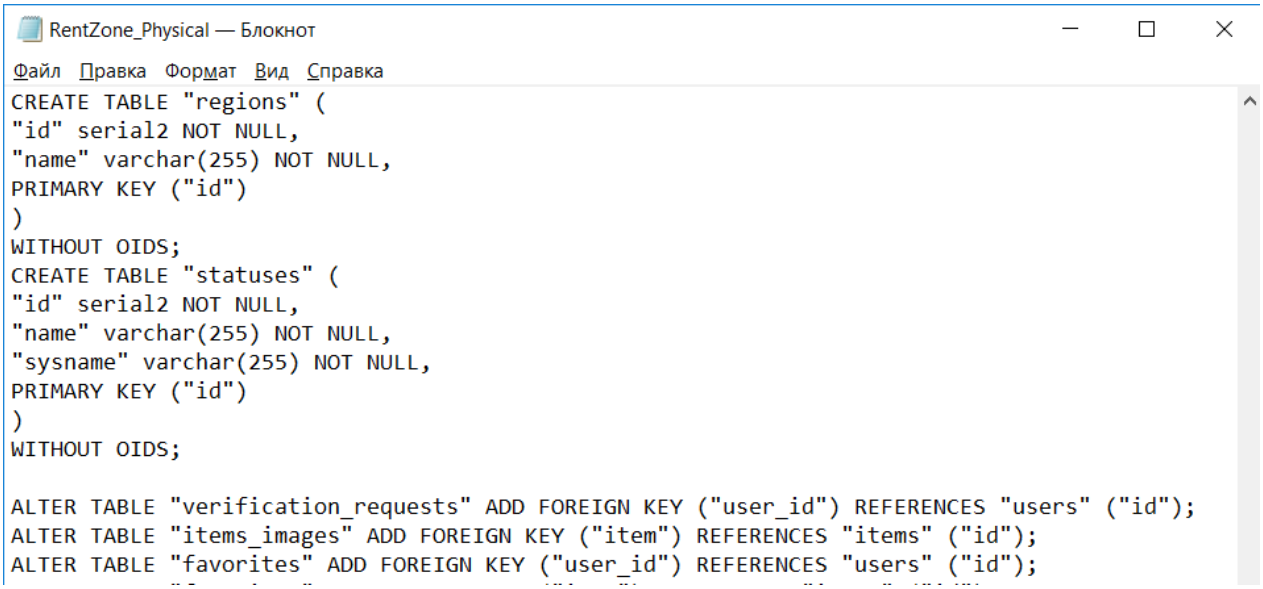

Рисунок 10 – SQL-скрипт

С помощью данного скрипта можно запросто создать базу данных для web-сервиса.

2.3 Обоснование выбора программных средств

Для разработки серверной части приложения в качестве языка программирования был выбран язык Java, среда разработки – IntelliJ IDEA, СУБД PostgreSQL, а также для удобства работы с базой данных был использован PgAdmin III.

Java – сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems [15]. Основными принципами языка являются:

– простота, объектная ориентированность и понятность;

– надёжность и безопасность;

– переносимость и независимость от платформы;

– высокая производительность;

– интрепретируемость, поточность и динамичность.

Java впервые выпущен в 1995 г. и с того времени набрал огромную популярность. На данный момент он продолжает оставаться одним самых из популярных языков программирования не только за счет своих особенностей, но и потому, что на данном языке написано огромное количество приложений, которые повседневно используются и их нужно поддерживать. Переход на другие языки приведет к тому, что все эти приложения придется переписывать, что является невозможной задачей, да и на фоне новоиспеченных языков Java не сильно уступает в производительности и простоте.

К тому же новый язык не сможет похвастаться таким огромным набором различных библиотек и фреймворков. Поэтому данный язык еще долго будет занимать лидирующие позиции в рейтинге популярности.

Одной из лучших сред для разработки, поддерживающей Java, является IntelliJ IDEA. Она включает поддержку всех последних фреймворков и технологий, включающие умное автодополнение, инструменты для анализа качества кода, удобную навигацию, современные инструменты рефакторинга кода, механизмы интеграции со средой тестирования TestNG, JUnit и системами управления версиями, включая Git, Subversion, Mercurial и CSV, а также интеграцию с автоматизированными инструментами сборки и управления проектом, включая Maven, Gradle, Ant [16].

Благодаря используемым технологиям повышается скорость написания кода, а также его качество, что способствует росту продуктивности разработчика. К тому же среда поддерживает не только Java, а также Groovy, Scala, HTML, CSS, JavaScript, CoffeeScript, ActionScript, LESS, XML и много других языков, что позволяет в одной среде разрабатывать как серверную часть, так и клиентскую.

В качестве СУБД был выбран PostgreSQL, так как он является мощным инструментом при создании больших проектов и обладает множеством возможностей. Созданный с использованием объектно-реляционной модели, PostgreSQL поддерживает сложные структуры и широкий спектр встроенных и

определяемых пользователем типов данных [17]. Поддерживает такие «фичи», как оконные функции, объединение запросов и Common Table Expressions (CTE) с использованием выражения WITH. Так же он обеспечивает расширенную ёмкость данных и заслужил доверие бережным отношением к целостности данных [18].

Работа с базой данных осуществлялась с помощью pgAdmin - это свободное кроссплатформенное программное обеспечение, предоставляющее графический интерфейс для работы с базой данных. Данное средство позволяет легко создавать, редактировать, вставлять данные в таблицу без написания SQL-запроса. А также содержит дополнительные возможности для создания SQL-запроса с помощью графического конструктора.

В качестве front-end языков был выбран стандартный набор языков, с помощью которых можно создать удобный и динамический web-сайт: HTML, CSS, JavaScript.

Связка HTML и CSS – это основа web-сайтов, с их помощью создается каркас страницы и добавляется внешний вид документа, в итоге получается страница, которую мы видим в браузере [19].

JavaScript – это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах [20]. С помощью JS можно создавать, удалять, скрывать, изменять стили элементов, создавать реакцию на действия посетителя, обрабатывать перемещения курсора мыши, нажатие клавиш, выводить сообщения, отправлять запросы на сервера и загружать данные без перезагрузки страницы и еще многое.

Вывод по второму разделу

В данном разделе была разработана функциональная и информационная модель для web-сервиса по предоставлению товара в аренду, а также было описано и выбрано программное обеспечение, и языки программирования для работы с ним.

3 Реализация программного обеспечения

3.1 Разработка базы данных

При помощи утилиты pgAdminIII под управлением СУБД PostgreSQL была создана база данных web-сервиса для аренды товара – rent_zone.

При помощи ранее упомянутого Navicat Data Modeler был получен скрипт для создания базы данных. На рисунке 11 отображается окошко с выполненным SQL-запросом по созданию таблиц.

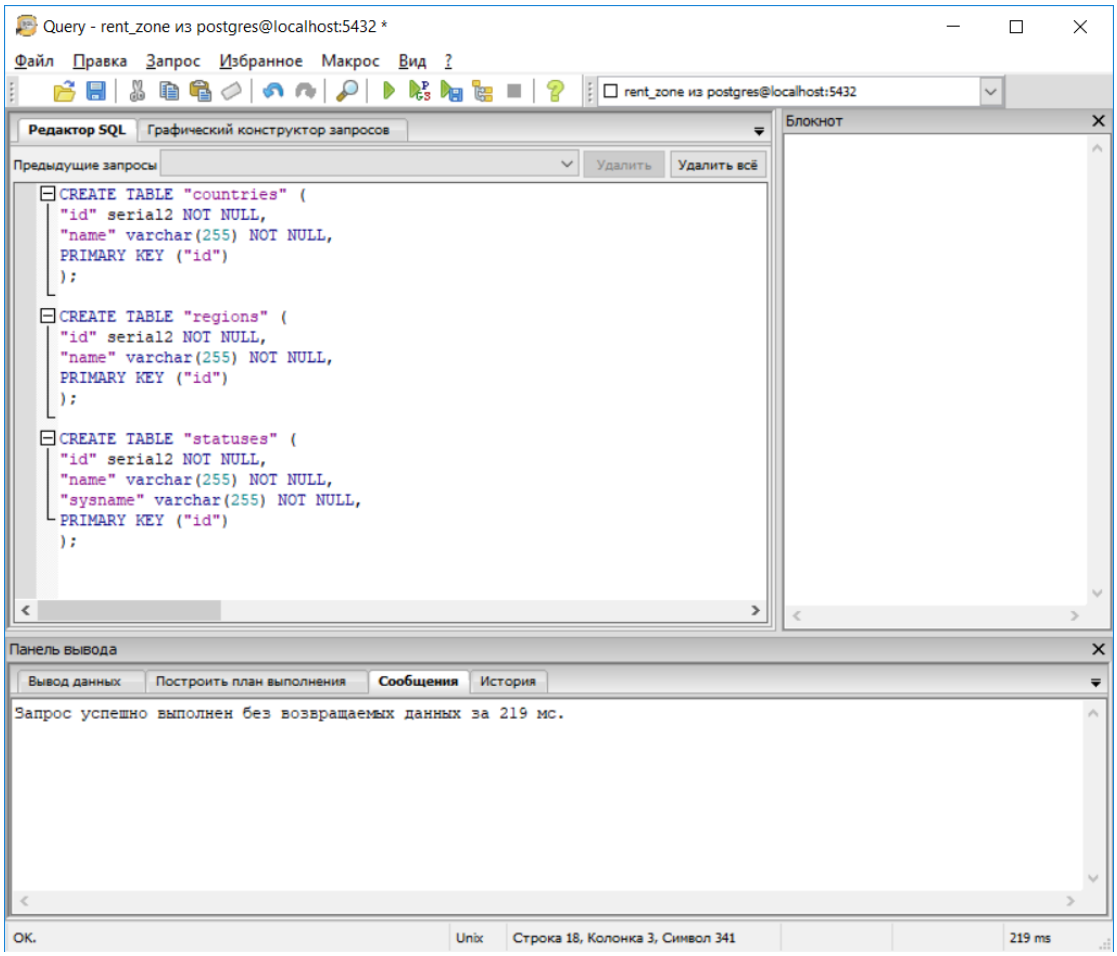

Рисунок 11 – Выполнение запроса на создание таблиц

В результате выполнение запроса получили 19 таблиц с ограничениями в виде первичного и внешних ключей представленные на рисунке 12. А также

были созданы 19 последовательностей, которые будут генерировать автоматически id для наших записей.

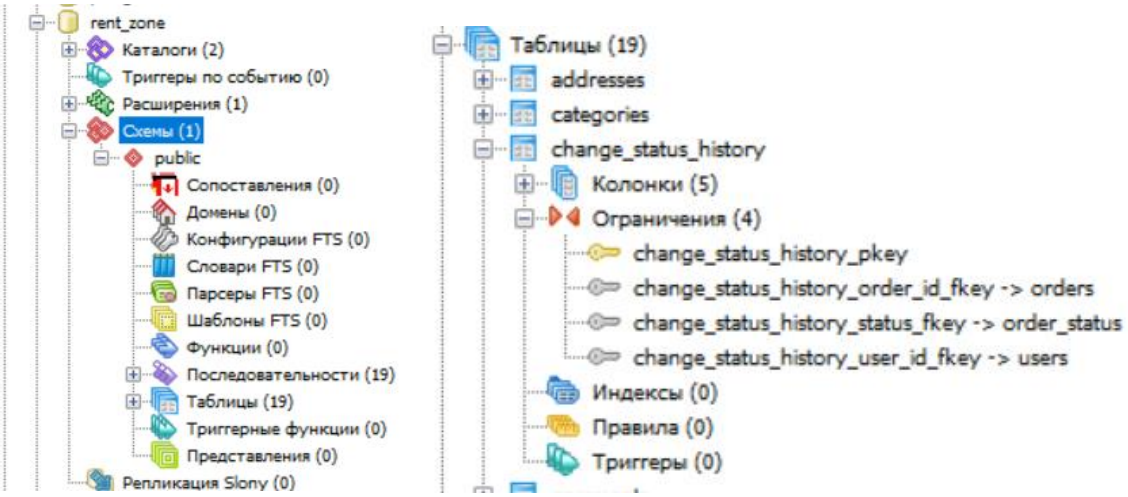

Рисунок 12 – Созданная база данных

Описание по каждой таблице представлено в таблице 5.

Таблица 5 – Описание таблиц базы данных

Название таблицы	Описание
items	Данные о товаре предоставляемого на аренду
orders	Данные о запросе на аренду товара
users	Данные пользователя
documents	Список документов
item_images	Фотографии товаров
delivery_methods	Список способов доставки
order_statuses	Статусы запроса на аренду
change_status_history	История изменения статусов запроса
document_types	Список типов документа
addresses	Список адресов

Продолжение таблицы 5

Название таблицы	Описание
countries	Список стран
regions	Список регионов
verification_requests	Запросы на верификацию пользователя
user_role	Роли пользователя
roles	Список ролей
favorites	Список любимых/желаемых товаров
categories	Список категорий
statuses	Список статусов товара
comments	Комментарии пользователей

Наполнение БД тестовыми данными являлось завершающим этапом разработки базы данных.

3.2 Разработка web-сервиса

Разработка приложение начинается с создание проекта при помощи ранее упомянутого средства IntelliJ IDEA.

В качестве сборщика проекта был выбран Maven. С помощью maven происходит загрузка в локальный репозиторий сторонних библиотек, компиляция, создания jar, создания дистрибутива программы, генерации документации [23]. Вся структура проекта описывается в файле pom.xml (POM – Project Object Model), в котором размещаются такие теги, как ptoject, groupId, artifactId, properties, dependencies, build и другие. Данные теги используется потом для сборки проекта в jar или другой указанный архив.

Для облегчения создания приложения воспользуемся готовым фреймворком – Spring Boot. Spring Boot – это полезный проект, целью которого является упрощение создания приложений на основе Spring [23]. Проще говоря, Spring представляет собой набор уже написанных классов и интерфейсов, которые помогут сильно облегчить создание приложения. Spring Boot же создан для управление всеми зависимостями, которые используются в Spring, автоматической конфигурацией, так же он встраивает web-сервер в наше приложение, что позволяет запускать приложение, как исполняемый jar-файл.

Для начала использования данного фреймворка, нужно добавить зависимости в наш pom-файл в соответствии с рисунком 13.

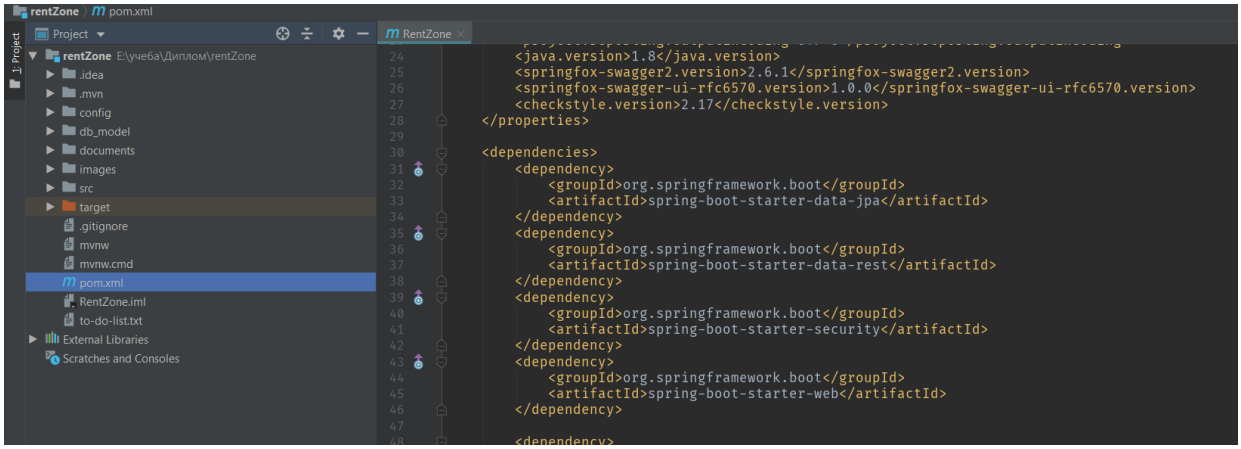

Рисунок 13 – Добавление Spring Boot зависимостей

Так как наше приложение будет использовать различные технологии, такие как JPA, Web, REST, нужно добавить соответствующие зависимости в проект.

Помимо перечисленных технологий была добавлена зависимость для разграничения доступа к выполнению определенных действий на сайте, поставляется вместе с spring-boot-starter-security.

В качестве архитектуры приложения был выбран MVC шаблон. Архитектура MVC предполагает разделение кода приложения на 3 части: Модель (Model), Вид или Представление (View) и Контроллер (Controller) [24].

Данная методика позволяет упростить большой по объему код, что помогает легко ориентироваться в коде и изменять его без ошибок.

В соответствии с данной архитектурой были созданы модели, которые соответствуют сущностям в таблице. Создание сущности «Пользователь» представлено на рисунке 14.

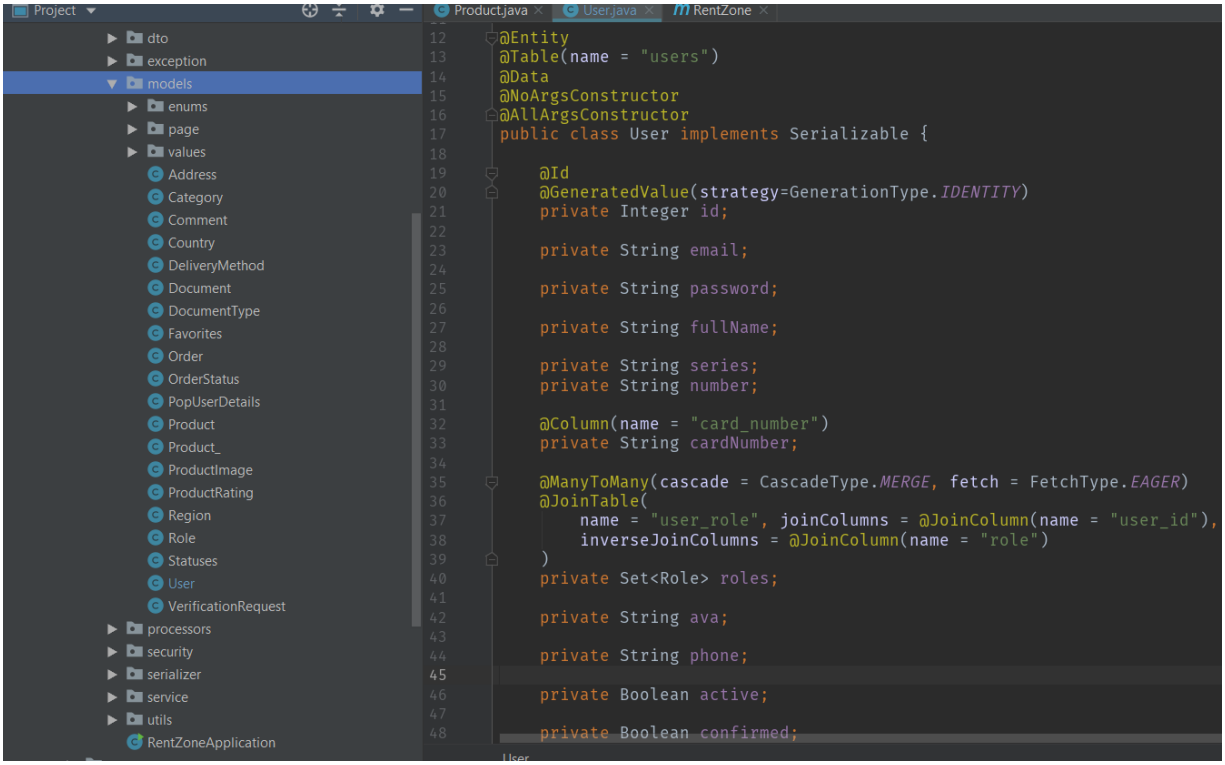

Рисунок 14 – Создание моделей

@ИмяАннтоации – специальная форма синтаксических метаданных, которая может быть добавлена в исходный код. Аннотация определяет необходимую информация для компилятора/интерпретатора; даёт информацию различным инструментам для генерации другого кода, конфигураций и т. д. [25].

С помощью аннотаций, представленных на рисунке 15, мы сопоставляем нашу модель с таблицей users. Таким образом наше приложение будет знать какая колонка в таблице будет соответствовать полю из класса, что позволит получать объекты из строк в таблице или сохранять их в качестве строки. Данную возможность предоставляет JPA технология [26].

Следующим этапом является создание Data Access Object (DAO) слоя – это слой для получения доступа к базе данных. На этом уровне происходит подключение к базе, выполнение запросов и конвертация результата в запрашиваемый java объект.

Для установки соединения приложения с базой данных была подключена зависимость postgresqsl. Подключение данной зависимости показано на рисунке 15.

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.1.1</version>
</dependency>
```

Рисунок 15 – Добавление зависимости для взаимодействия с базой данных

На рисунке 16 представлен класс для получения данных пользователя.

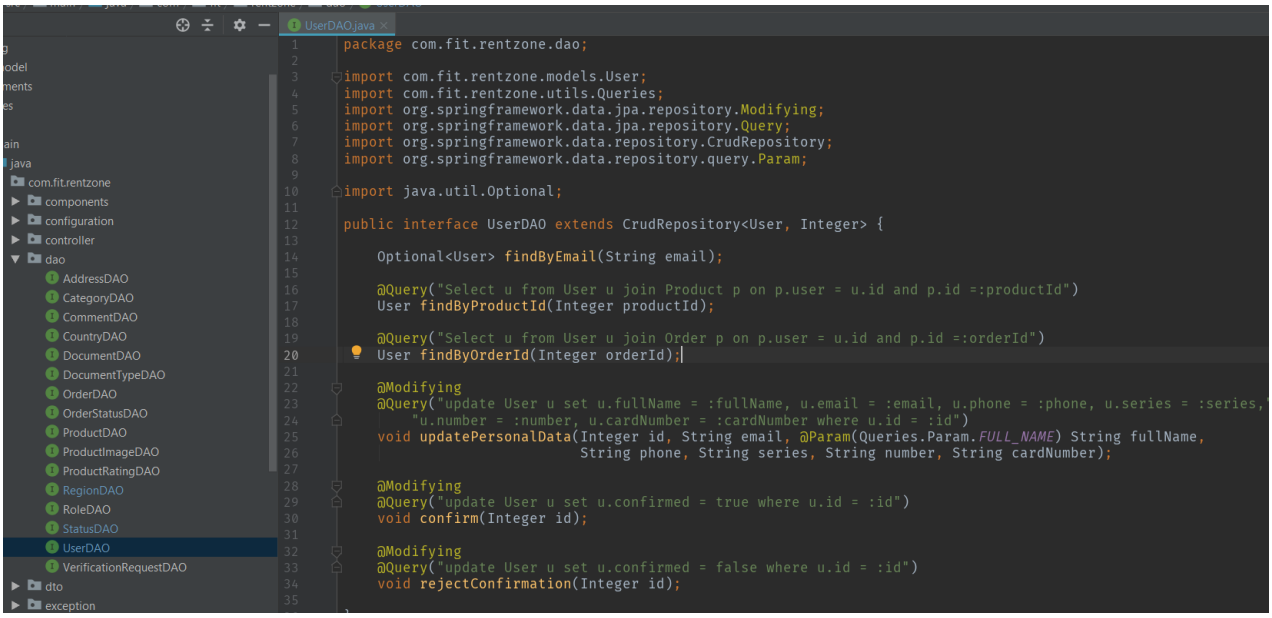

Рисунок 16 – Класс UserDAO

Здесь явно не прописано подключение к базе и реализация метода. Всё это спрятано в интерфейсе CrudRepository, который предоставляет нам Spring фреймворк. Благодаря технологии Spring Data большинство методов реализовано за нас. Он обеспечивает реализацию основных операций, таких как поиск, сохранение, удаление данных (CRUD операции).

Если же предоставленных методов недостаточно, то можно расширить базовый интерфейс для своей сущности, дополнив его своими методами. Особенностью данной технологии является составление запроса из имени метода. Для этого используется механизм префиксов findBy, readBy, queryBy, countBy, и getBy, далее от префикса метода начинает разбор остальной части. К примеру метод с названием findByEmail преобразуется в запрос «select * from users where email = ?». При более сложных запросах можно использовать аннотацию Query, где явно прописать sql-скрипт.

Чтобы все вышеперечисленные запросы смогли выполниться, должно произойти подключение к базе. Spring автоматически установит соединение нужно только указать url базы, username и пароль к базе. Все свойства для подключения прописываются в специальном файле-конфигурации, откуда Spring достанет значения и использует при подключении. Файл с конфигурацией представлен на рисунке 17.

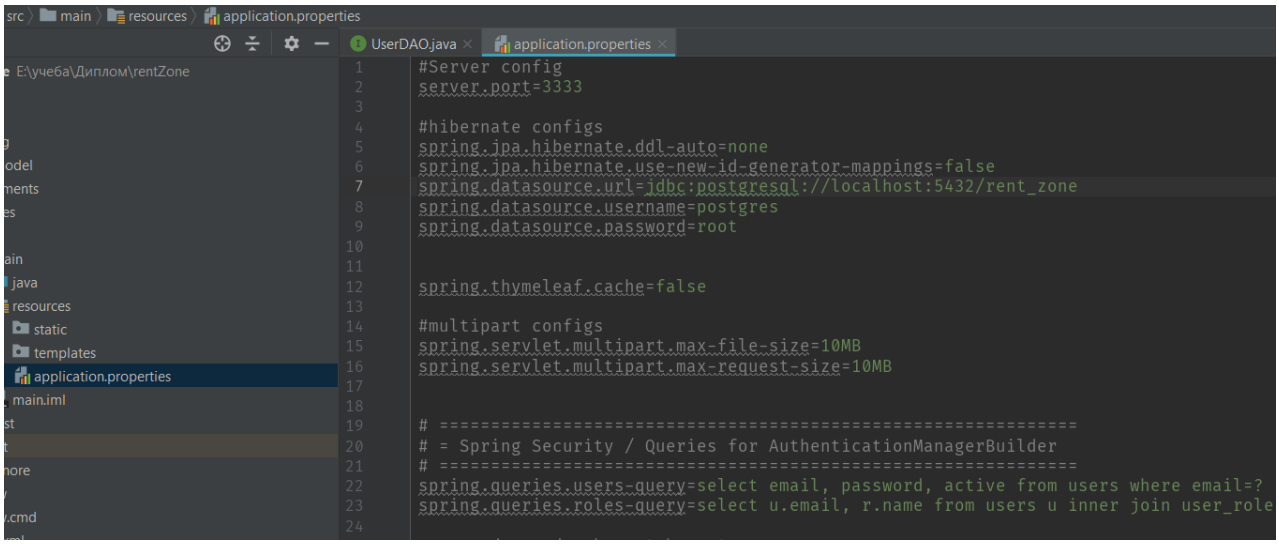

Рисунок 17 – Файл с конфигурацией

Следующая задача – создание сервисов. В данном слое прописывается основная бизнес-логика приложения. В сервисе используются DAO классы или другие сервисы. Хорошим тоном в java является использование интерфейсов, которые описывают поведение класса. На рисунке 18 представлен интерфейс сервиса для юзеров.

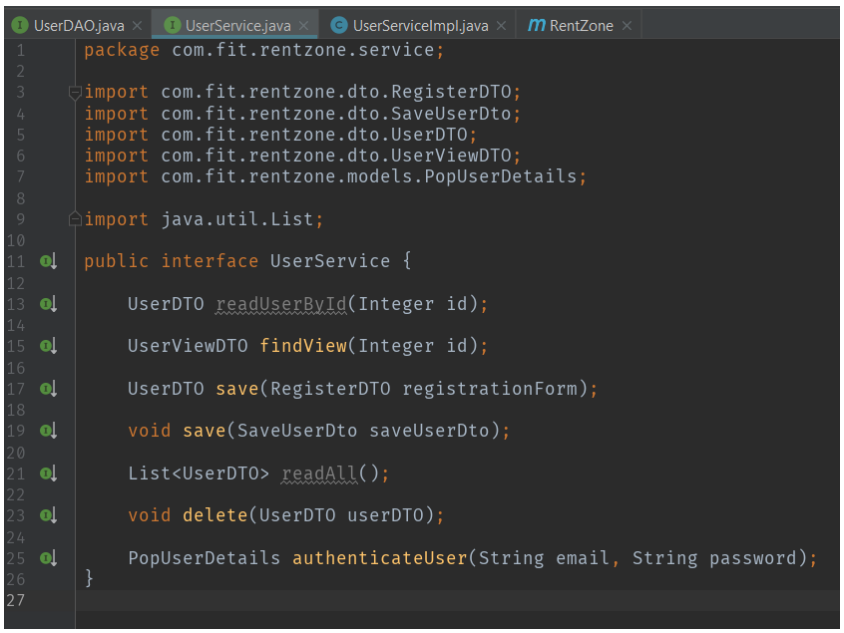

Рисунок 18 – Интерфейс UserService

Интерфейс помогает отобразить саму суть класса, каким поведением он будет обладать [27]. Реализация интерфейса представлена на рисунке 19.

Рисунок 19 – Реализация интерфейса UserService

Последнем слоем в серверной части являются Controller-классы. Класс, отвечающий за работы с сущностью «Пользователь», представлен на рисунке 20.

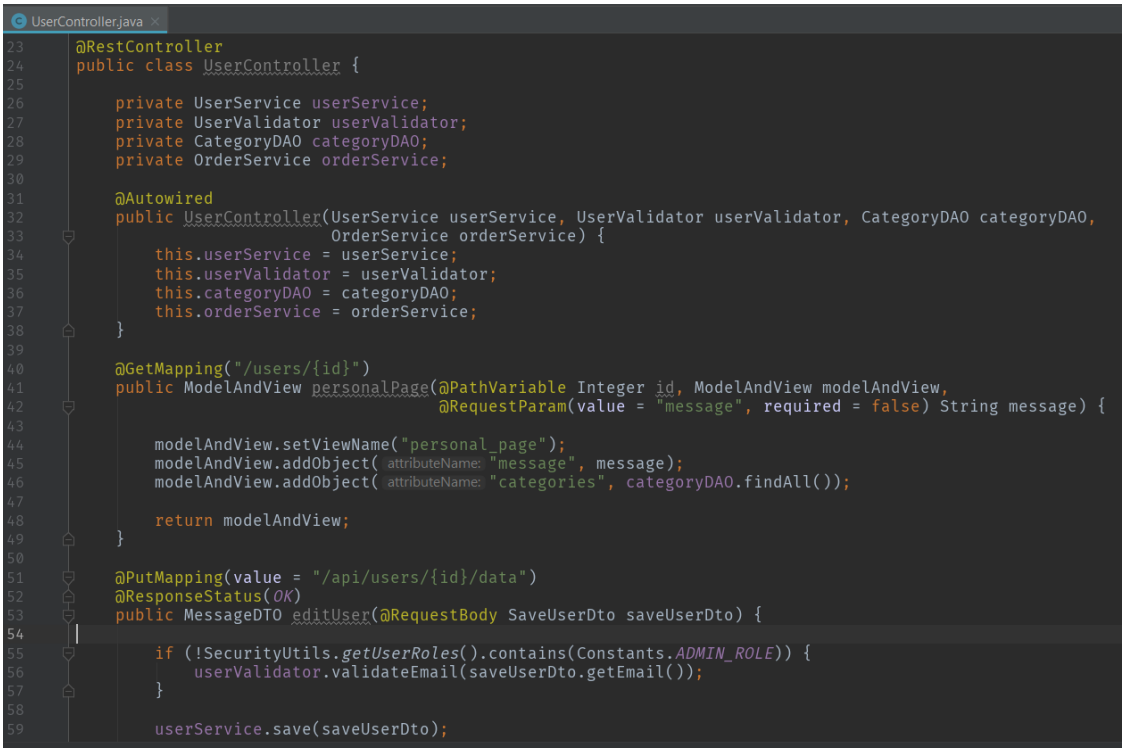

Рисунок 20 – Класс UserController

Контроллер – это класс, предназначенный для непосредственной обработки запросов от клиента и возвращения результатов. Чаще всего, в самом контроллере не описывается логика обработки данных. Задача методов в контроллере – корректно принять данные, вызвать сервисные методы в нужном порядке и корректно возвратить результат [28].

В качестве результата контроллер может вернуть отдельную модель, DTO (Data Transfer Object) или представление.

View (Представление, Вид) отвечает за отображение данных модели, – как правило, генерируя HTML, которые мы видим в своём браузере. В качестве построителя HTML был выбран thymleaf, так как он подходит для генерации View (View Layer) Web-приложения основываясь на структуре MVC, а также поддерживает полностью интеракцию с Spring Framework [29].

На рисунке 21 представлен html-код странички юзера с использованием thymleaf вставок.

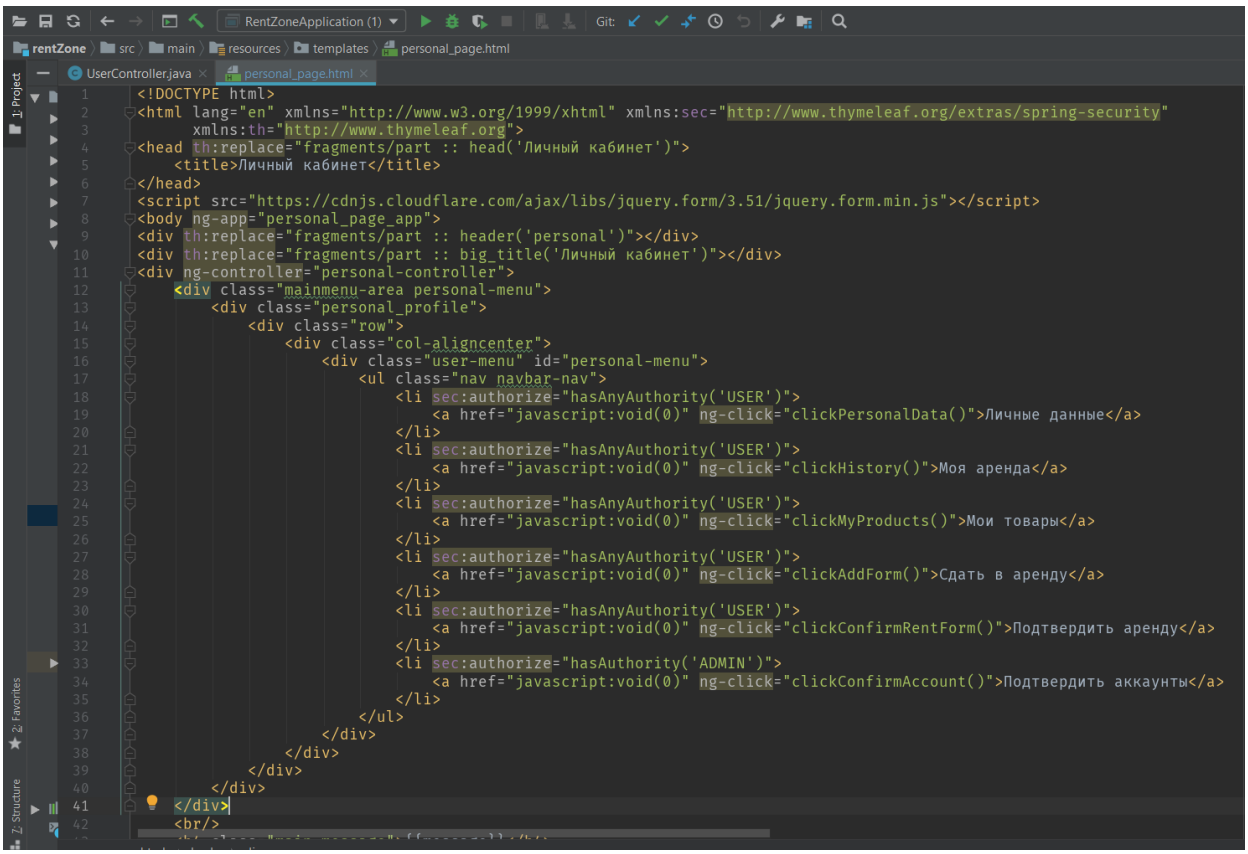

Рисунок 21 – Html-код страницы пользователя

С помощью thymleaf можно автоматически генерировать html-код, который подставит в нужно место значения, полученные из контроллера. К тому же можно скрыть некоторый функционал, если роль пользователя, не соответствует требуемой и еще много других возможностей.

3.3 Разработка интерфейса АИС

Важнейшим аспектом в web-приложении является его интерфейс. Многие могут заметить, что если их не привлекает дизайн сайта или сайт не очень удобен в использовании, то они не только не будут частыми клиентами, а и вовсе сразу же покинут сайт, не оценив его внутренности. Поэтому второй большой задачей в разработке любого web-приложения является создание привлекательного, интуитивно понятного интерфейса, который будет завлекать посетителей своей простотой и ясностью.

Первым делом была создана главная страница web-сервиса, которая представлена на рисунке 22.

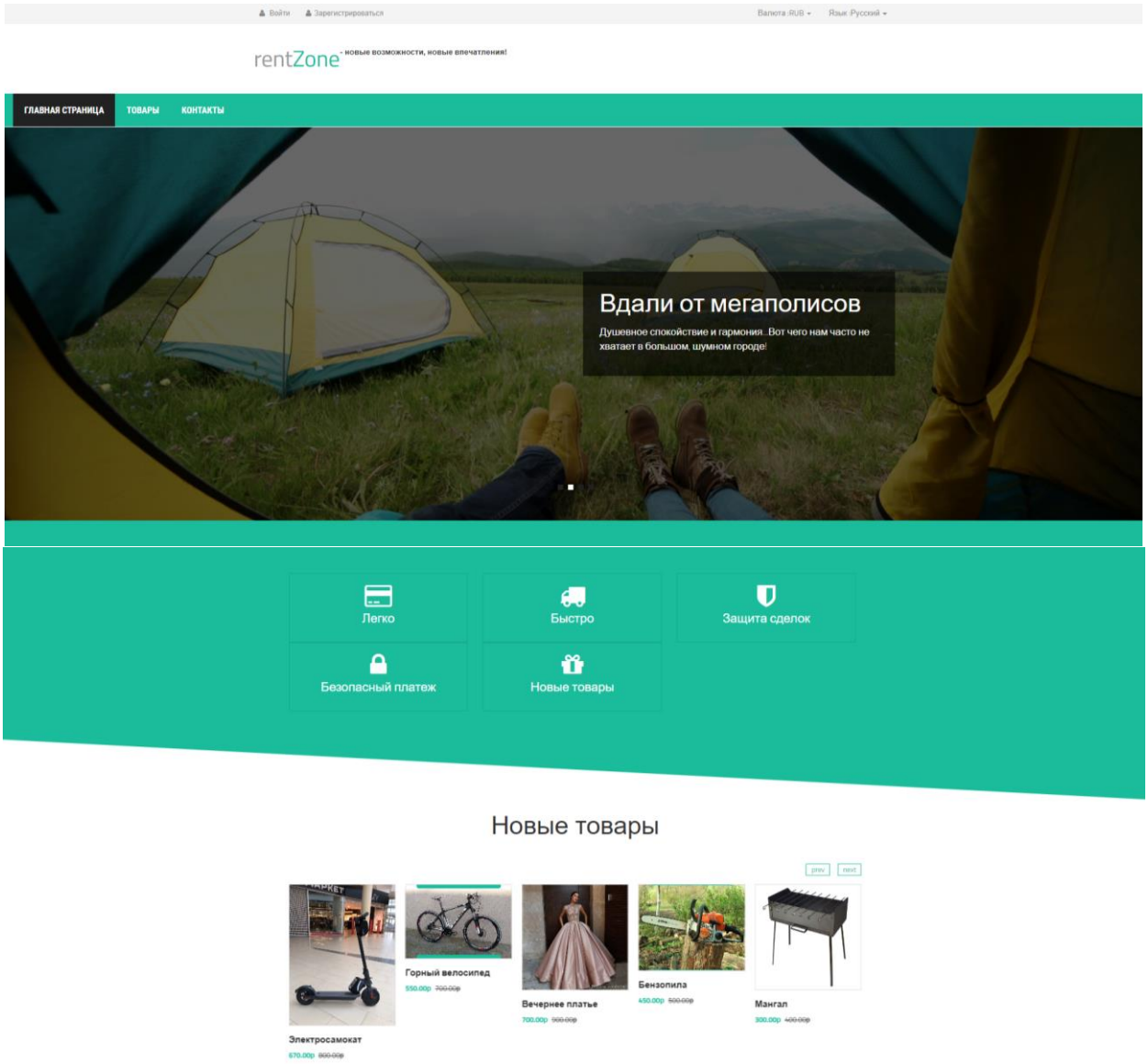

Рисунок 22 – Главная страница rentZone

В верхней части сайта можно найти кнопки «Войти» и «Зарегистрироваться». Меню сайта состоит из «Главной страницы», «Товары» и «Контакты». Далее следует слайдер с фотографиями, который должен заинтересовать пользователя и способствовать появлению желания отправиться в путешествие, посетить театр и т.д., что возможно приведет к воспользованию услуг web-сервиса по аренде товара. Далее размещен блок с новыми товарами и быстрым переходом к ним. Ниже располагается информация о работе сайта и подвал с описанием предназначения web-

сервиса, навигацией и быстрым переходам к товару с выбранной категорией. Данная часть сайта представлена на рисунке 23.

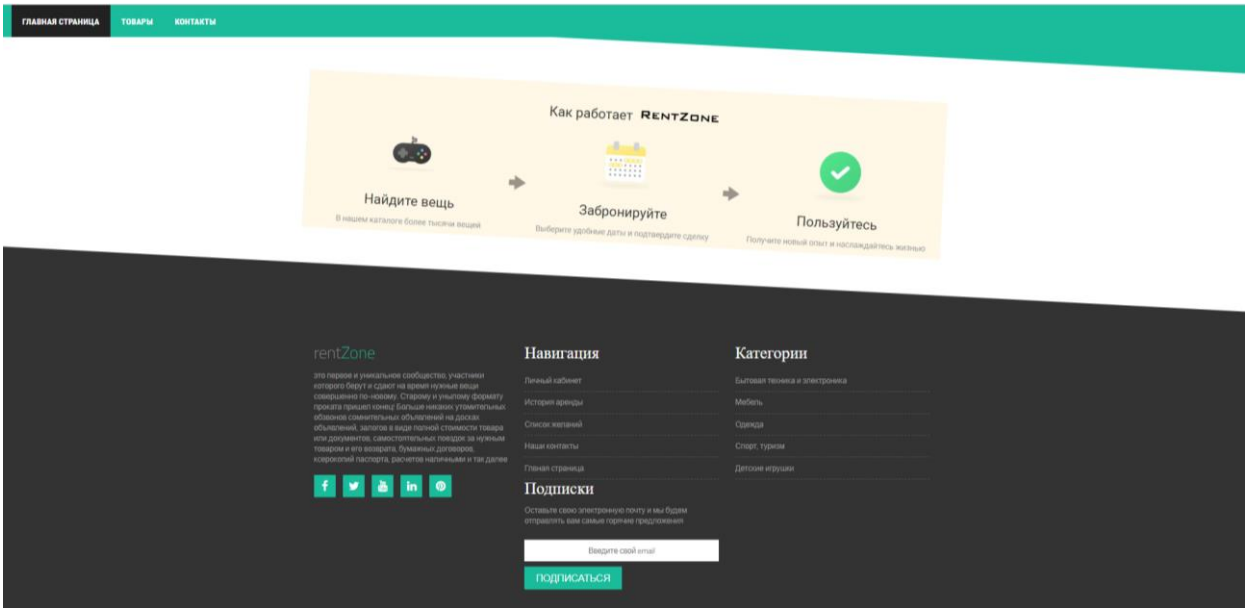

Рисунок 23 – Нижняя часть главной страницы

Дальше была реализована форма для авторизации, которая отображается на рисунке 24.

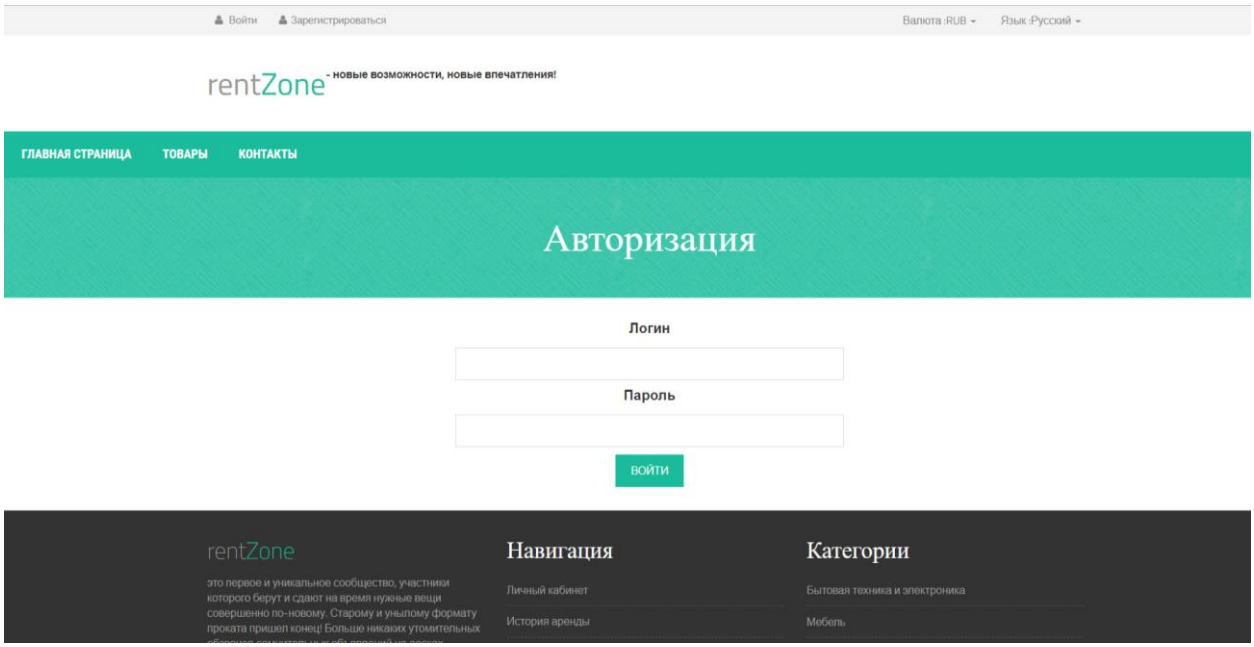

Рисунок 24 – Форма для авторизации

На рисунке 25 представлена форма для регистрации.

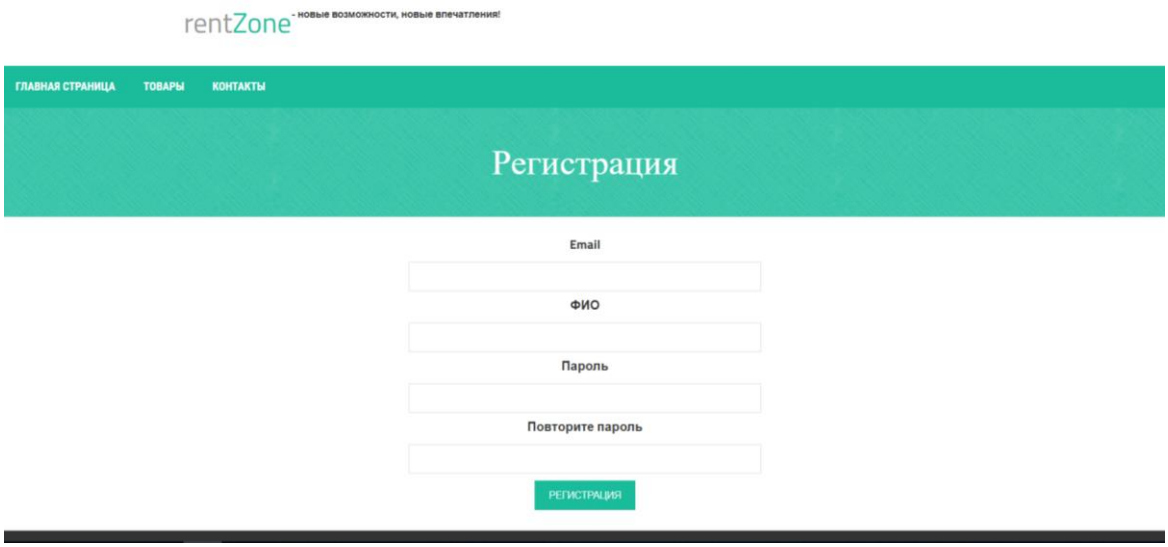

Рисунок 25 – Регистрационная форма

Так как пользователей часто ошибается при вводе информации, были реализованы механизмы для недопущения этих ошибок. На рисунке 26 представлен неверный ввод email-а пользователя и сообщение о допущении ошибки.

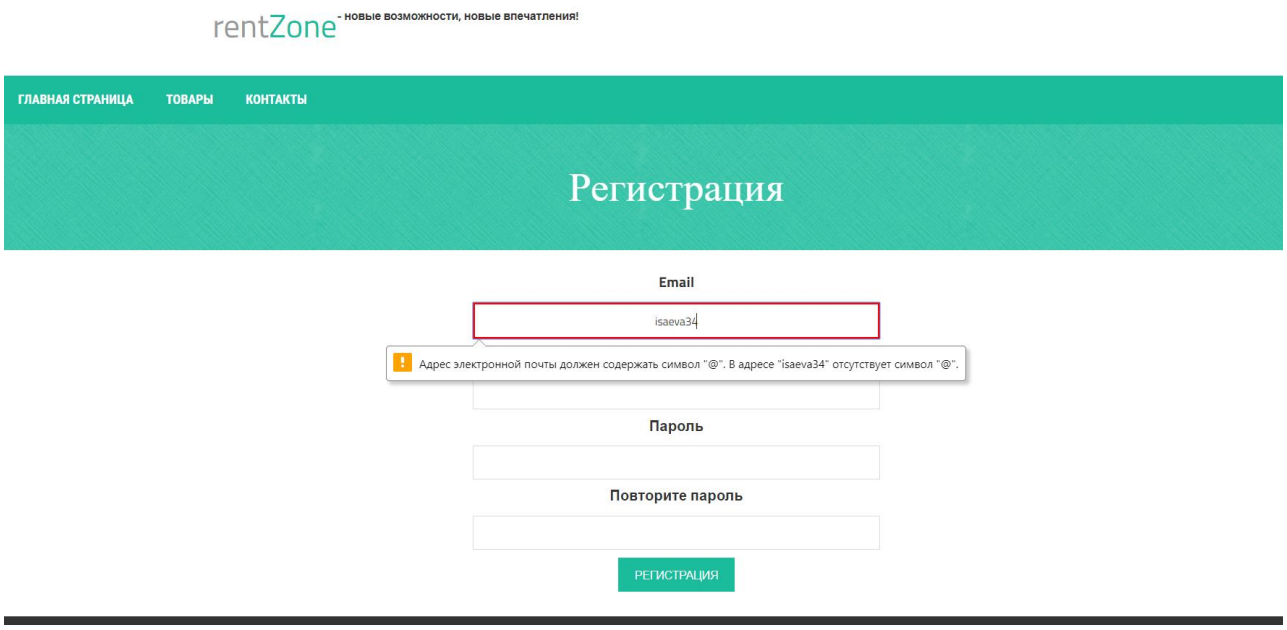

Рисунок 26 – Результат валидации

При успешной авторизации у пользователя становятся доступными кнопки для перехода в личный кабинет и выход с профиля, представленные в правой стороне на рисунке 27.

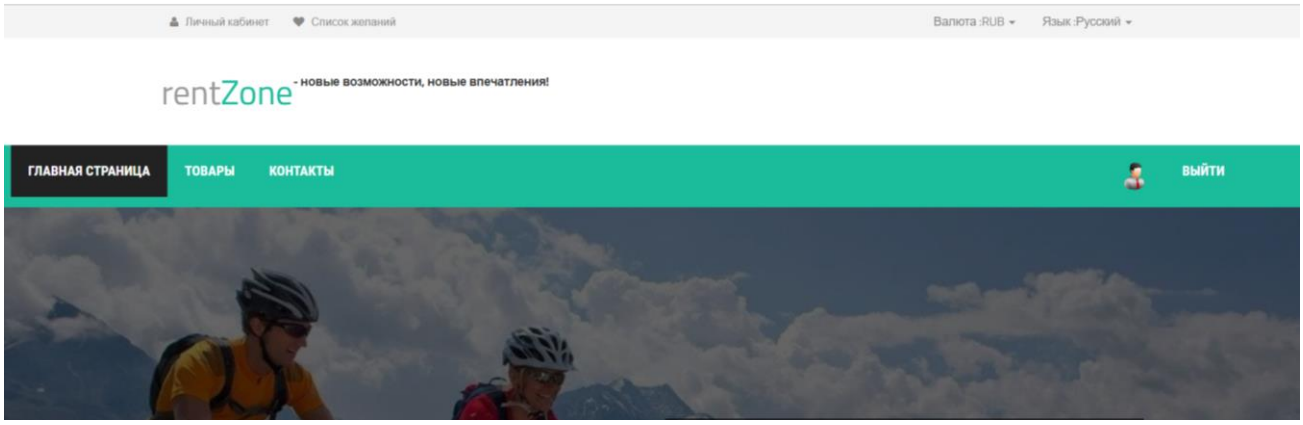

Рисунок 27 – Главная страница после авторизации

При нажатии на иконку человечка мы попадем в личный кабинет авторизованного пользователя.

Личный кабинет пользователя представлен на рисунке 28.

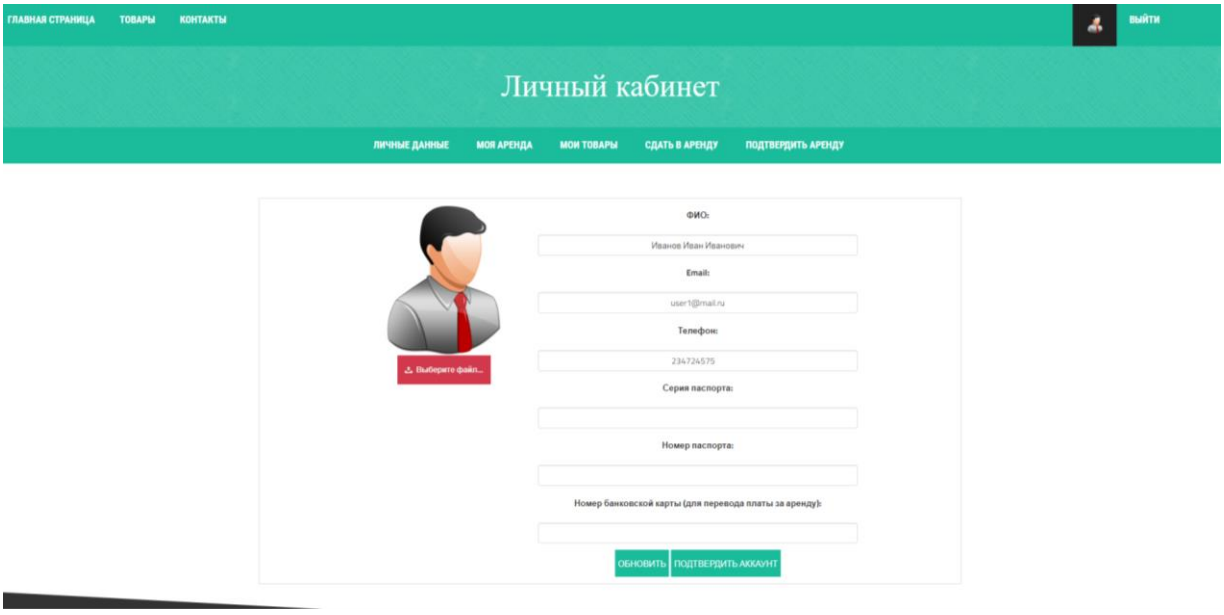

Рисунок 28 – Личный кабинет

В личном кабинете находятся такие вкладки, как «Личные данные», «Моя аренда», «Мои товары», «Сдать в аренду», «Подтвердить аренду». При открытии личного кабинета сразу открывается вкладка «Личные данные», где пользователь может ввести информацию, которая будет использоваться в ходе предоставления аренды/аренды товара.

Вкладка «Сдать в аренду» представлена на рисунке 29.

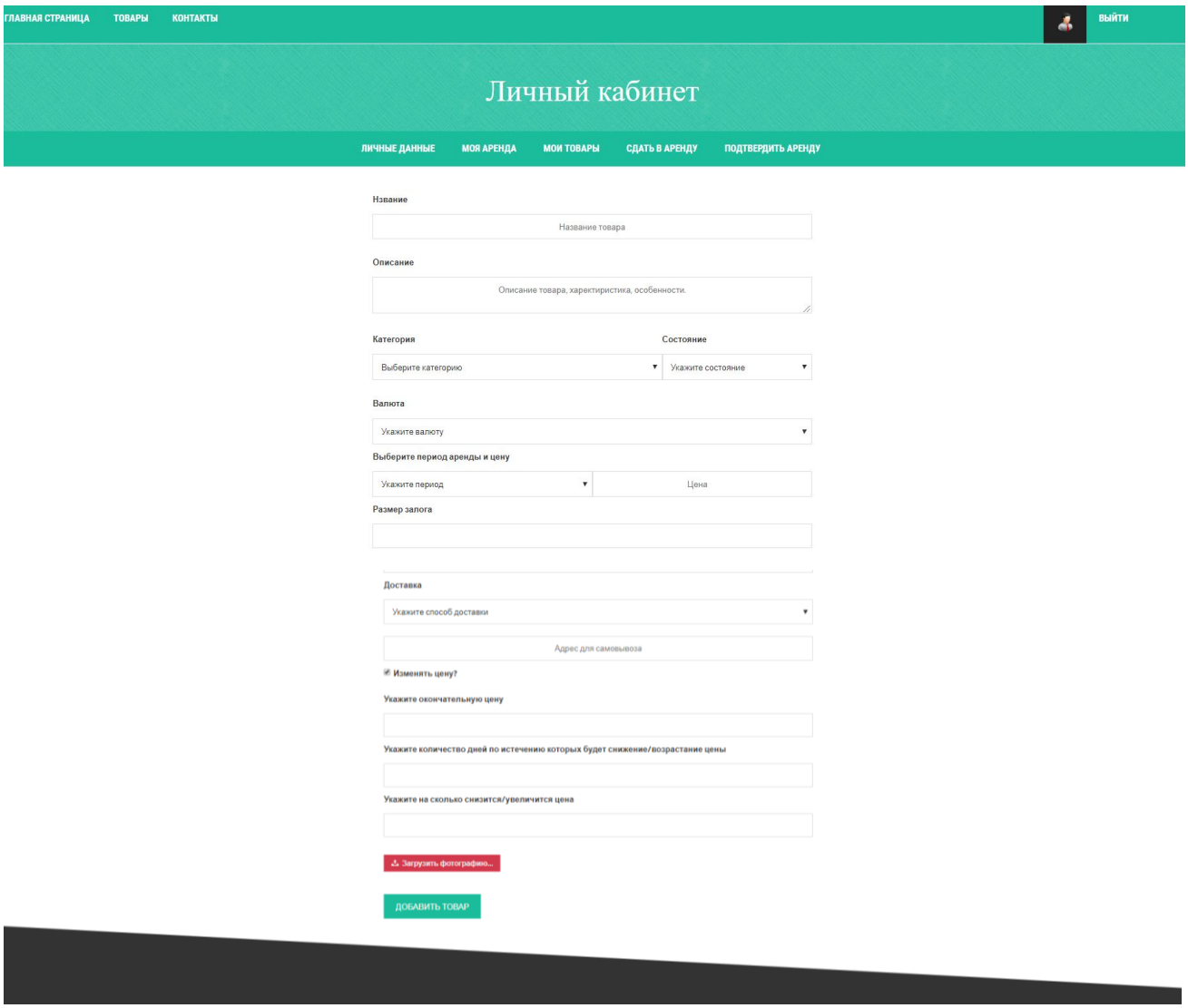

Рисунок 29 – Вкладка для добавления товара

Для добавления товара можно указать следующую информацию: название, описание, категория, состояние, валюта, цену за определенный период, размер залога, доставка и адрес. Так же можно изменять цену со временем. Для этого нужно указать окончательную цену, количество дней, по истечению которых будет увеличение/уменьшение цены и количество, на которое изменится цена. При достижении указанной окончательной цены, изменение цены приостанавливается. Так же можно загрузить фотографии товара. Далее отправленная форма проходит валидацию. На рисунке 30 показана форма, которая не прошла валидацию.

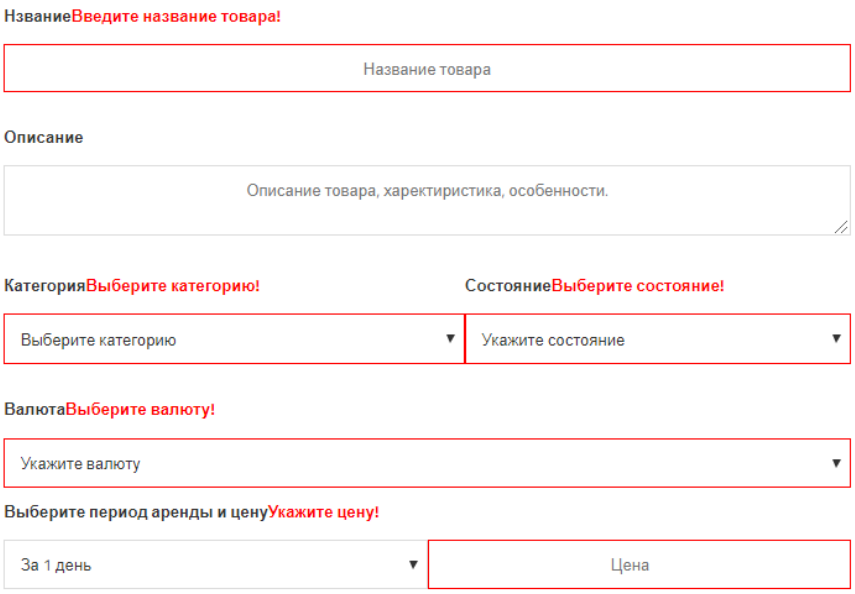

Рисунок 30 – Валидация при добавлении товара

На рисунке выше подсвечены красным обязательные поля для заполнения. Валидация формы реализована с помощью JS и представлена в листинге 1.

Листинг 1 – Валидация формы для добавления нового товара

```
function validateForm(form) {
   var isValid = true;
   if(form['name'] === "") {
      isValid = false;
      setErrorMessage("#product_name_label", "Введите название товара!")
   }
   if (form['state'] === "") {
      isValid = false;
      setErrorMessage("#product_currency_label", "Выберите валюту!")
   }
   if (form['category'] === "") {
      isValid = false;
      setErrorMessage("#product_category_label", "Выберите категорию!")
   }
   if (form['price'] === "") {
      isValid = false;
      setErrorMessage("#product_price_label", "Выберите категорию!")
   }
   if (form['currentPrice'] < 0) {
      isValid = false;
      setErrorMessage("#product_price_label", "Цена должна быть больше 0!")
   }
   return isValid;
}
```

Чтобы подсветить красным и вывести сообщения с подсказками, была реализована функция, представленная в листинге 2.

Листинг 2 – Добавления сообщения с подсказкой

```
function setErrorMessage(labelId, message) {
    var label = $(labelId);

    $('<span/>', {
        'text': message,
        'class': 'error_message'
    }).appendTo(label);

    var labelForAttr = label.attr('for');

    $('#' + labelForAttr).css('border-color', "red");
}
```

Если все поля валидны, то текст с подсказками убирается и товар успешно добавляется.

На вкладке «Мои товары», представленной на рисунке 31, располагаются все товары, которые добавил пользователь.

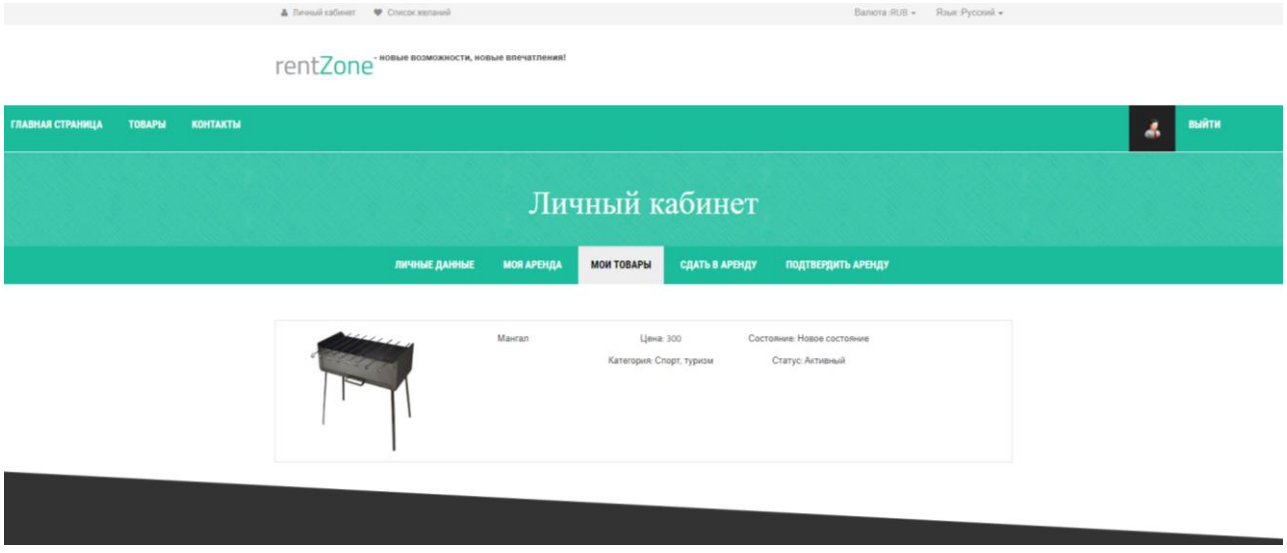

Рисунок 31 – «Мои товары»

На рисунке выше представлен один товар с основной информацией. Статус товара показывает его активность. Статус «Активный» означает, что товар свободен и его можно взять в аренду, «Неактивный» – то, что на данный момент товар в пользовании.

Для редактирования товара нужно нажать на окно с товаром и изменить данные в полях, которые представлены на рисунке 32. Удаление товара находится рядом с кнопкой «Обновить».

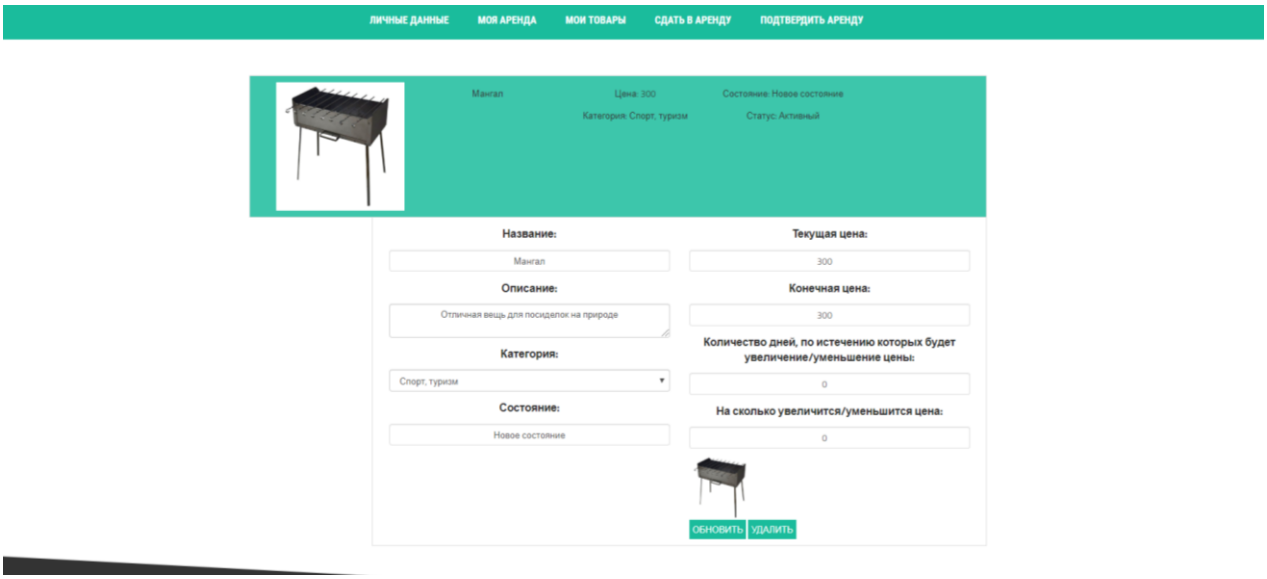

Рисунок 32 – Редактирование товара

Вкладка «Моя аренда» и «Подтвердить аренду» заполнена в случае, если вы запросили товар в аренду или ваш товар был выбран для аренды.

На рисунке 33 представлена вкладка «Подтвердить аренду».

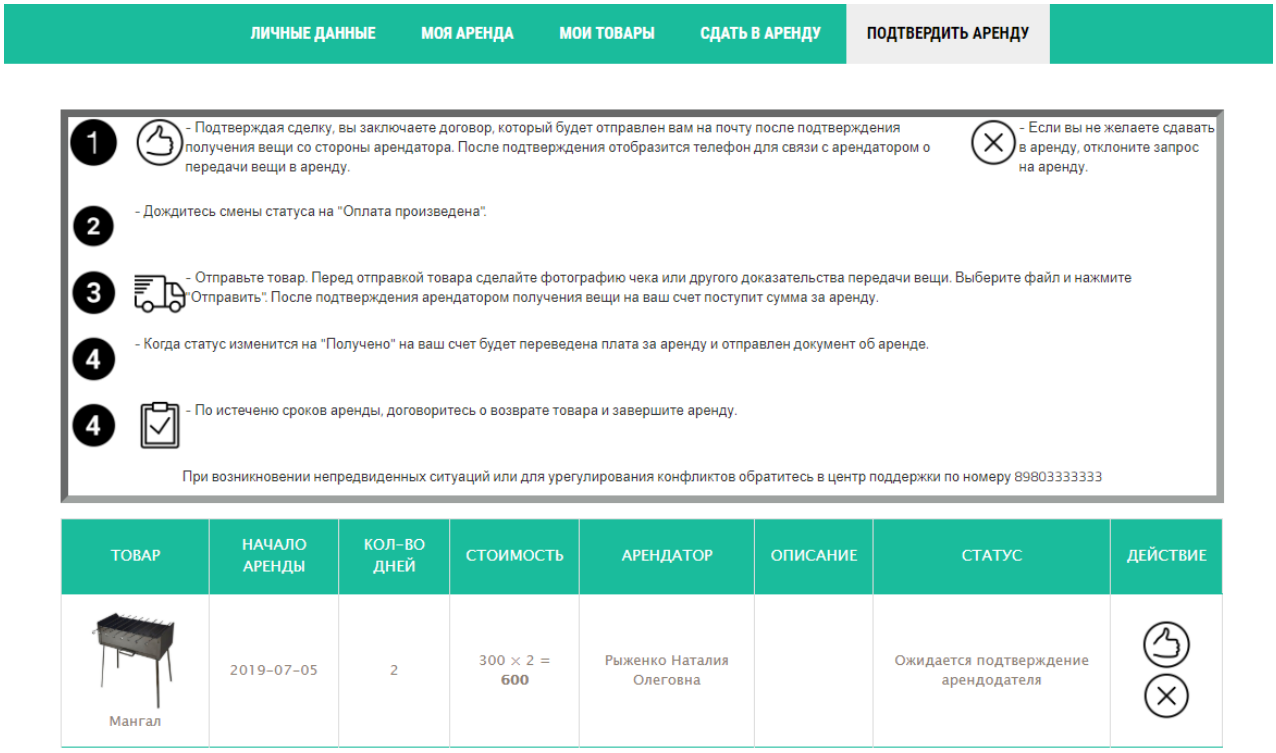

Рисунок 33 – Вкладка «Подтвердить аренду»

На рисунке выше представлена инструкция, как подтвердить аренду. Это позволит новым пользователям всегда перед глазами иметь краткое руководство с описание каждого шага, что также позволит понять, как будет проходить процесс аренды. Ниже инструкции располагается таблица с информацией о запросах: название товара, который будет арендован, дата начала аренды, срок аренды, стоимость, ФИО арендатора, описание, текущий статус запроса и действия. Аналогичный интерфейс имеет вкладка «Моя аренда» только со своими инструкциями для арендатора.

Раздел с товарами, предоставляющими в аренду представлен на рисунке 34.

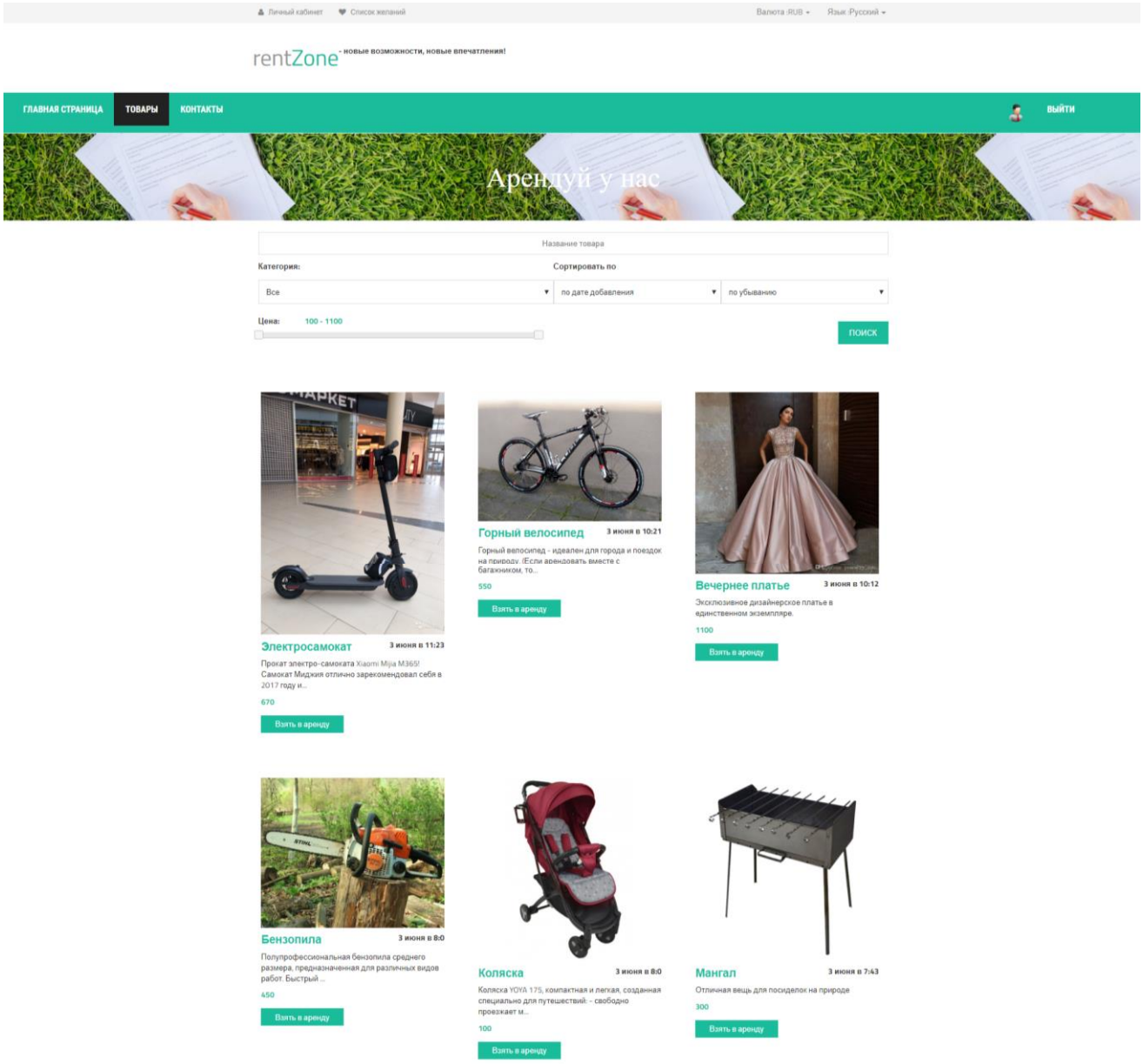

Рисунок 34 – Просмотр товаров

На данной странице реализован просмотр товаров, поиск, фильтрация и сортировка для удобного поиска вещи потенциальным арендатором.

Чтобы узнать подробную информацию о товаре, прочитать отзывы, нужно нажать на название товара. В результате откроется страница для просмотра товара, представленная на рисунке 35.

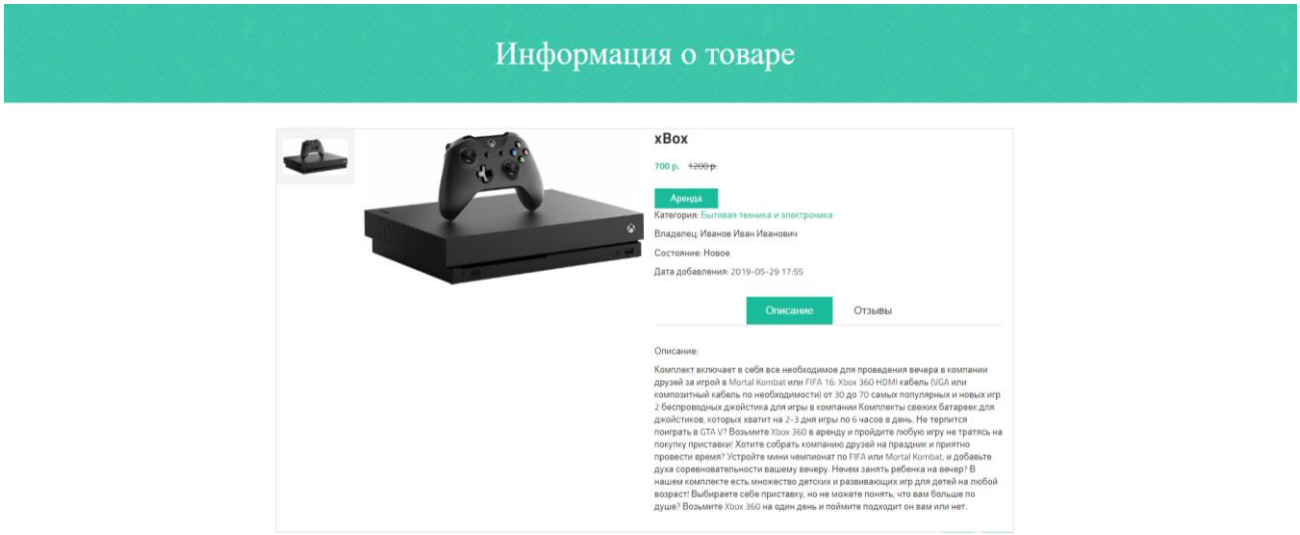

Рисунок 35 – Информация о товаре

Для того, чтобы взять товар в аренду, служит кнопку «Аренда», которая приведет вас к заполнению необходимых данных арендатора в соответствии с рисунком 36.

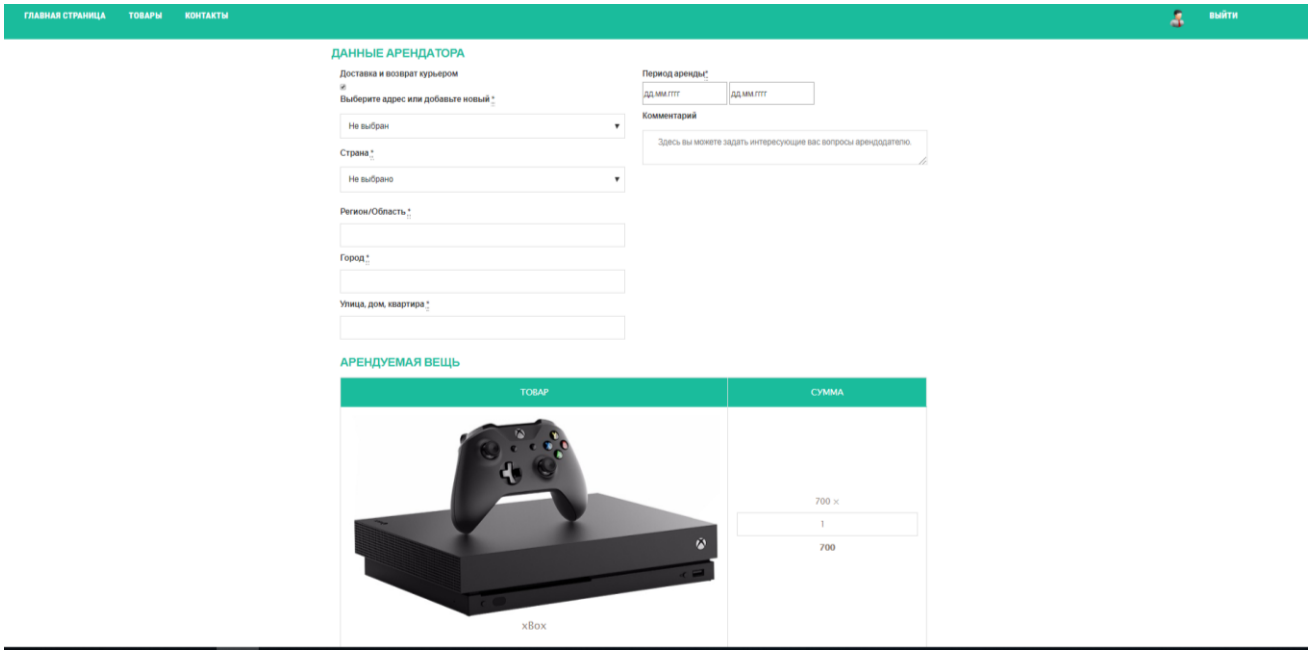

Рисунок 36 – Аренда товара

После заполнения всех необходимых данных создастся запрос на аренду товара и запрос отобразится в личном кабинете в разделе «Моя аренда».

3.4 Тестирование и отладка программного продукта

Тестирование программного обеспечения является неотъемлемой частью создания ПО и служит для проверки соответствия между реальным и ожидаемым поведением программы, осуществляемое на конечном наборе тестов, выбранном определенным образом.

Начнем тестирование со страницы каталога товаров. Главная страница и страница товаров должны быть доступны всем авторизированным и не авторизированным пользователям.

При переходе на «Товары» у неавторизированного пользователя не запрашивается авторизация, а свободно можно просматривать и искать товары в соответствии с рисунком 37.

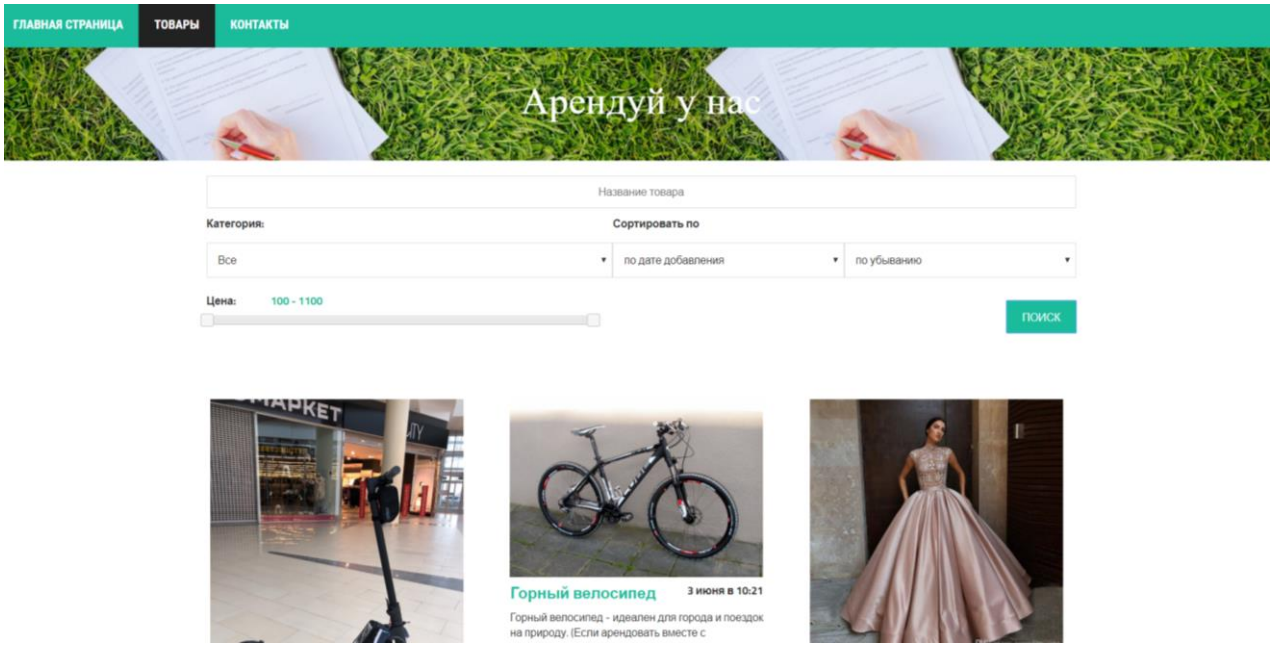

Рисунок 37 – Просмотр товаров неавторизированным пользователям

Введем любые две буквы и выставим сортировку по цене по возрастанию. На рисунке 38 отображается работа поиска.

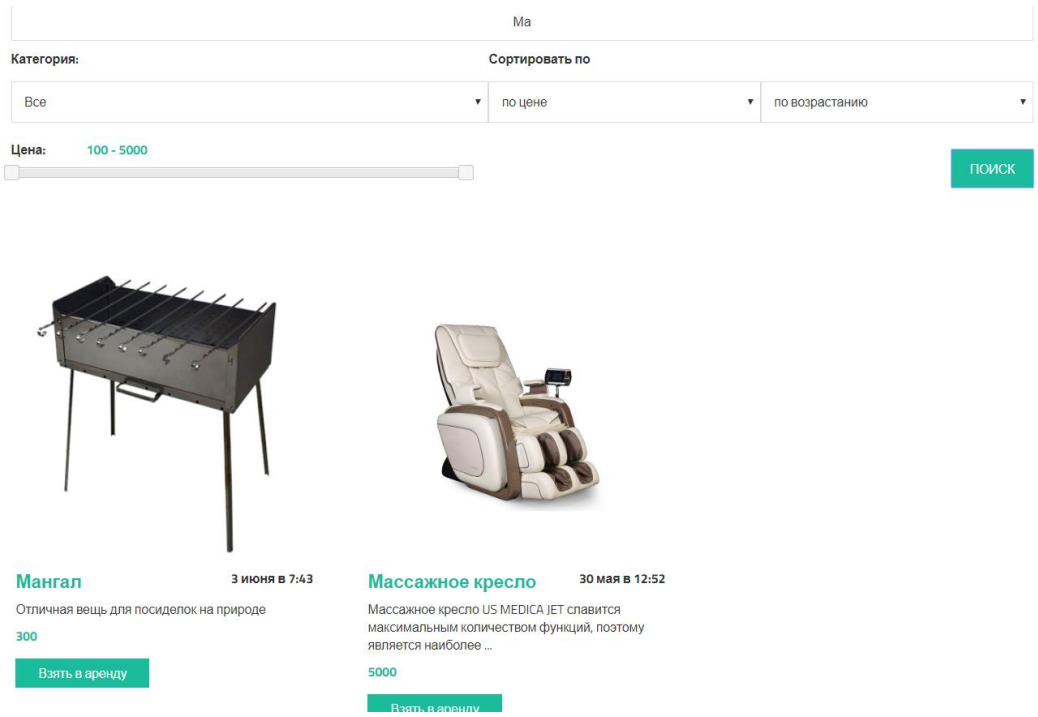

Рисунок 38 – Работа поиска

Убедимся в правильной работе выбора диапазона цены и категории товара. Зададим цену от 660 до 844 и выберем категорию «Бытовая техника и электроника» и получим результат, представленный на рисунке 39.

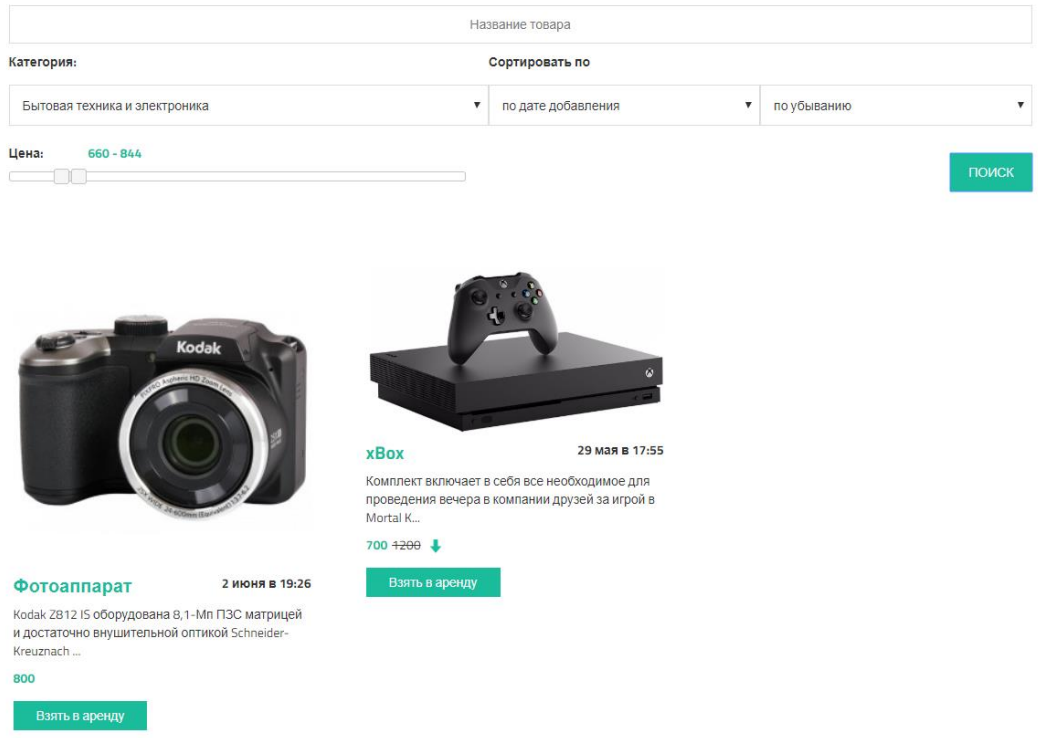

Рисунок 39 – Работа поиска

Для просмотра товара нажмем на название товара и осуществляется переход на следующую страницу, отображенную на рисунке 40.

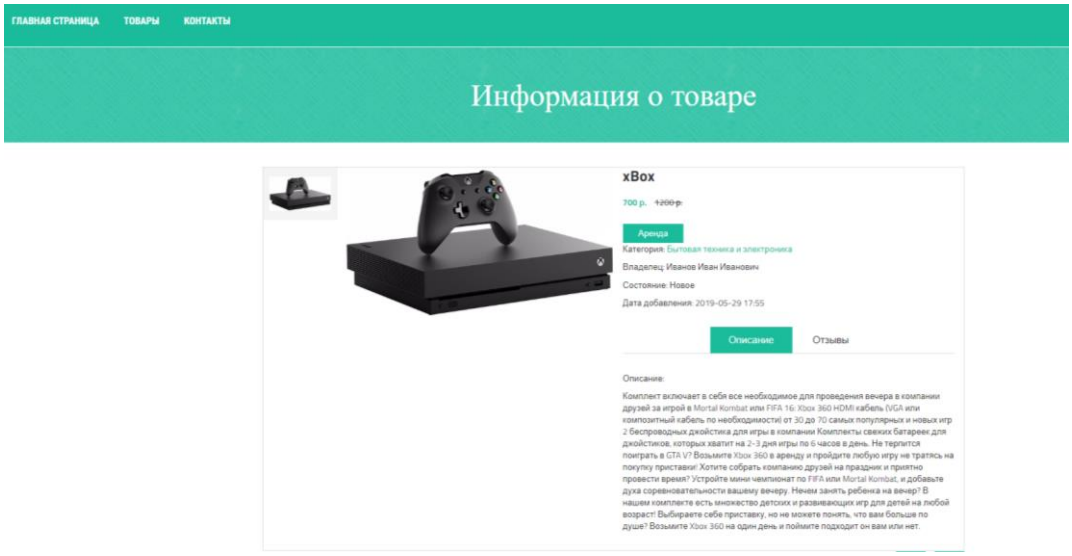

Рисунок 40 – Информация о товаре

Функционал по аренде товара должен быть открытым только зарегистрированным и авторизированным пользователям. При нажатии на кнопку «Аренда» срабатывает защита и осуществляется переход на страницу авторизации. Зарегистрируем пользователя и проверим валидацию данных при регистрации. В итоге будут подсвечиваться поля красным как показано на рисунке 41, если были введены неверные данные.

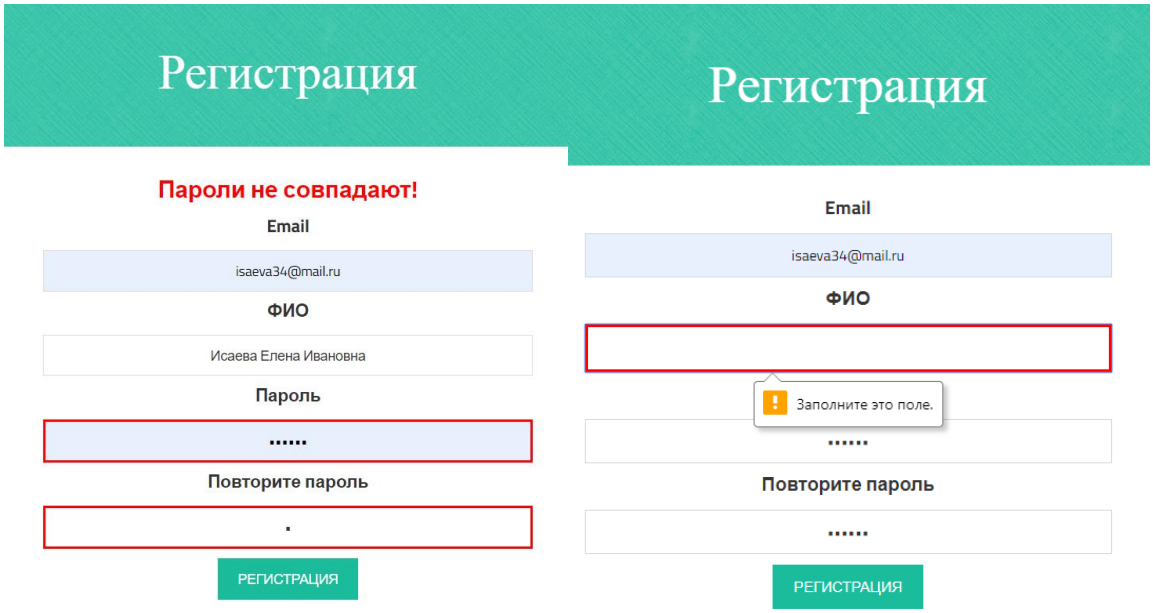

Рисунок 41 – Валидация данных

После успешной авторизации пользователь теперь может перейти на страницу аренды. Но так как при регистрации заполняются не все необходимые данные, то при попытке аренды, если пользователь не прошел подтверждение аккаунта, будет выдано сообщение как на рисунке 42.

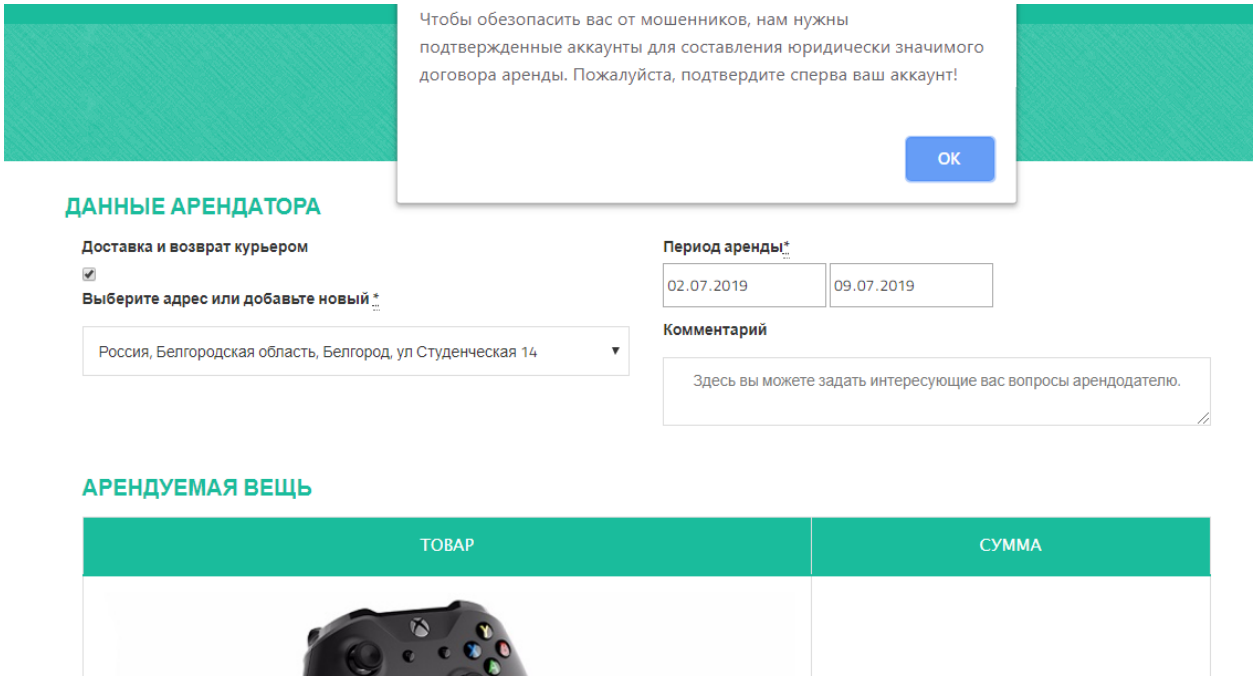

Рисунок 42 – Требование подтверждения аккаунта

Перейдем в личный кабинет и нажмем «Подтвердить аккаунт». В результате получим сообщение, представленное на рисунке 43.

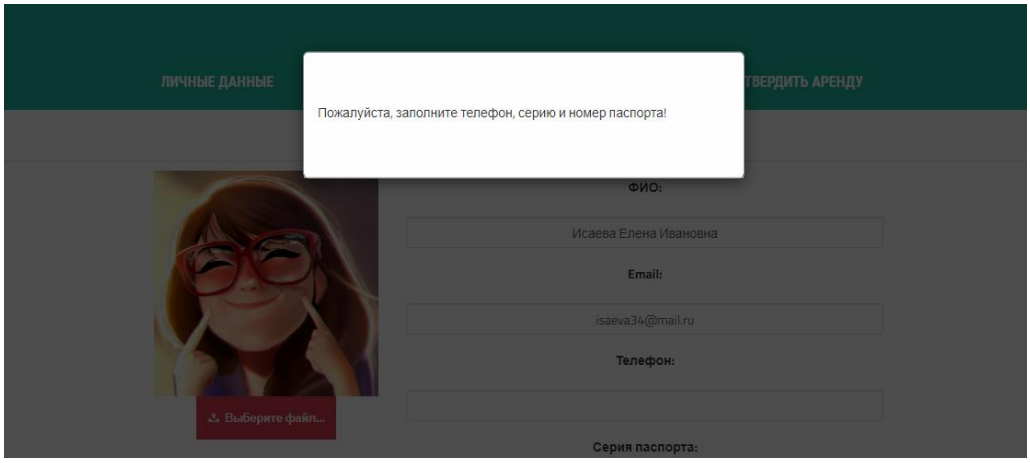

Рисунок 43 – Сообщение о заполнении полей

Обновим данные, заполнив телефон и данные паспорта. Обновление представлено на рисунке 44.

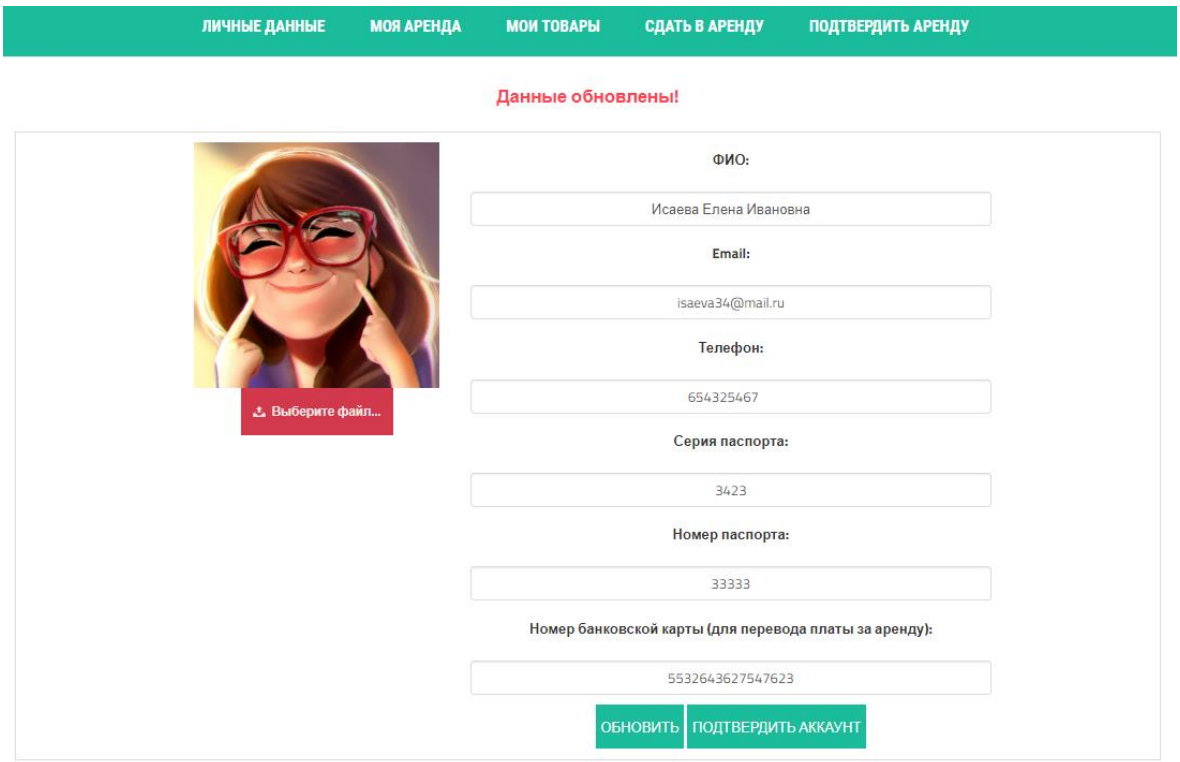

Рисунок 44 – Обновление данных

Опять пробуем подтвердить аккаунт. На этот раз мы получим окно для загрузки данных, необходимых для подтверждения аккаунта в соответствии с рисунком 45.

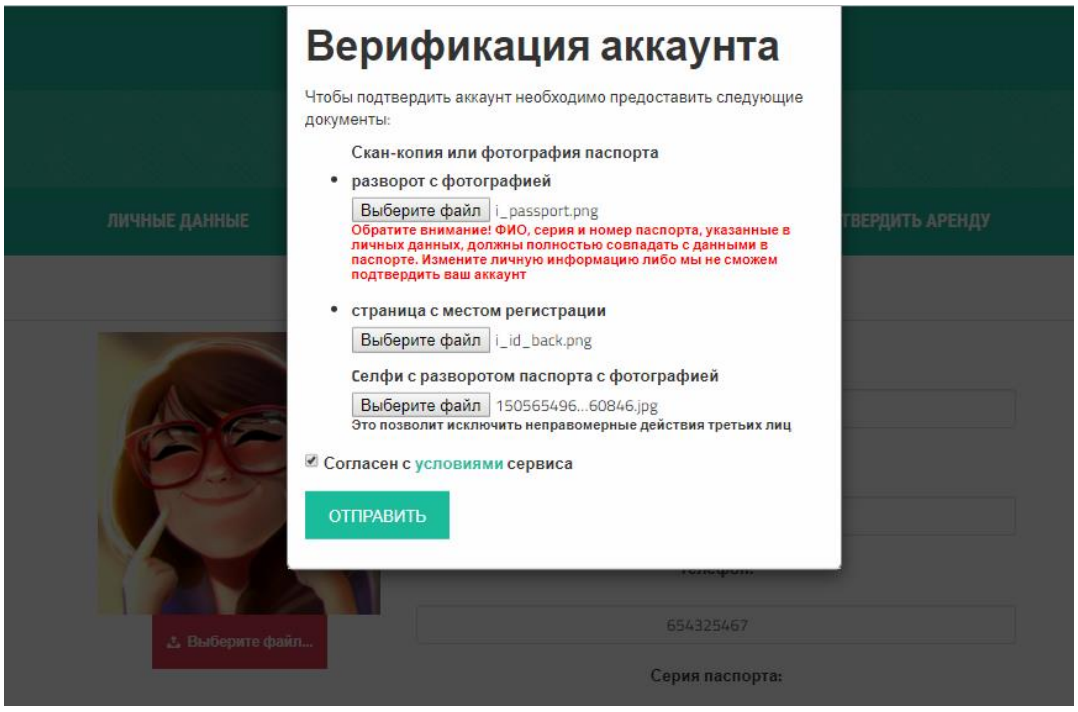

Рисунок 45 – Окно для загрузки паспортных данных

Отправляем данные и получаем сообщение об успешные отправки. Пока заявка не будет принята или отклонена администрацией, кнопка становится неактивной как показано на рисунке 46.

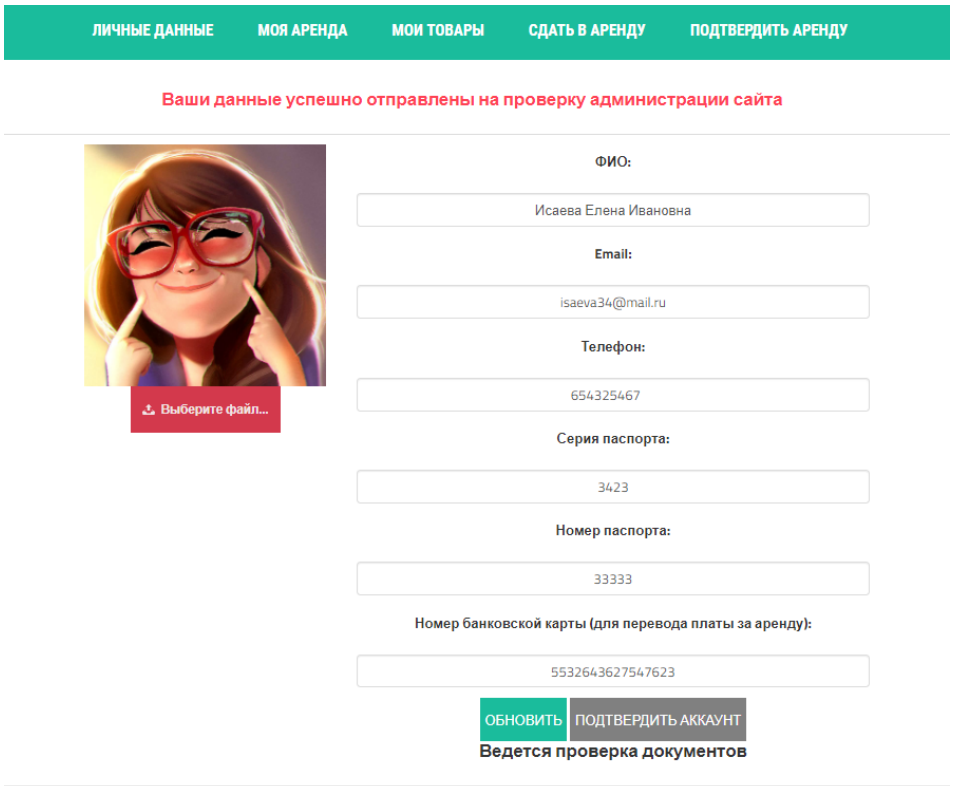

Рисунок 46 – Сообщение об успешные отправки данных

Администрация сайта просматривает заявки на соответствующей вкладке, представленной на рисунке 48.

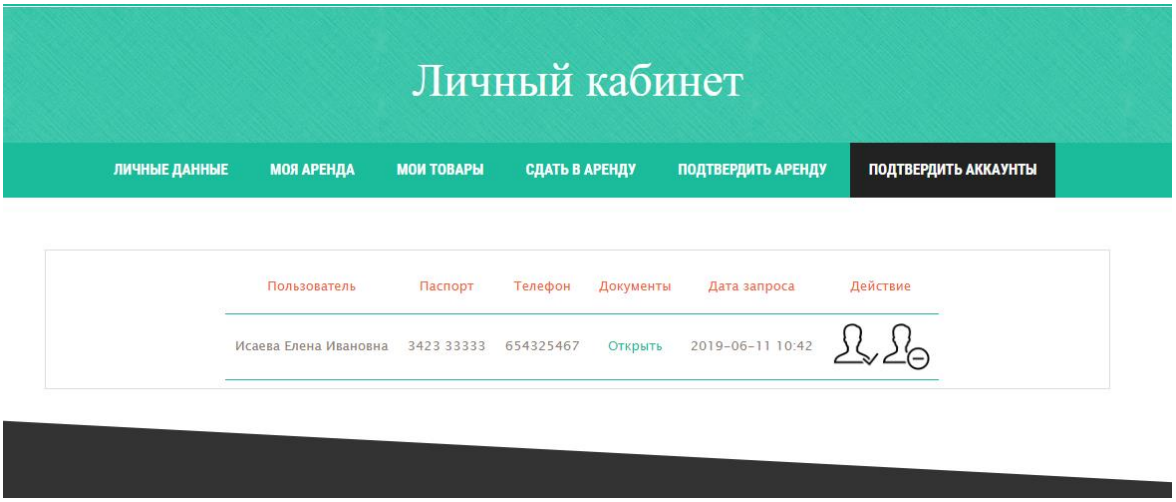

Рисунок 48 – Вкладка администрации

При нажатии на «Открыть» в столбце «Документы» открываются загруженные пользователем фотографии в соответствии с рисунком 49.

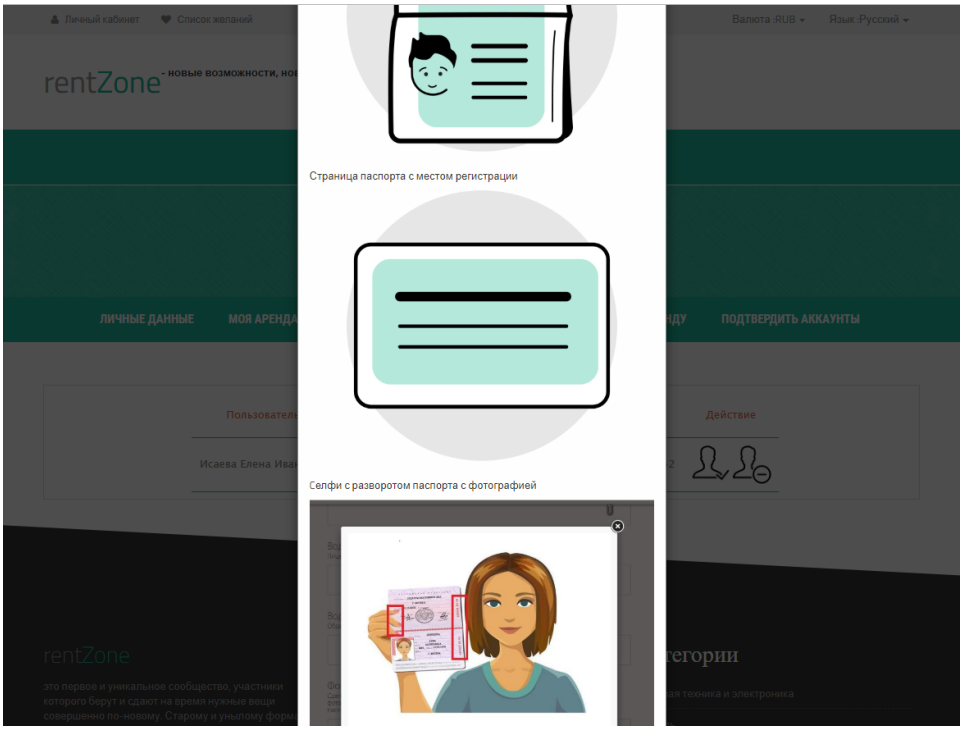

Рисунок 49 – Отправленные документы пользователя

Если все данные совпадают, то администрация подтверждает аккаунт, иначе подтверждение аккаунта отклоняется. Подтверждение/отклонение представлено на рисунке 50.

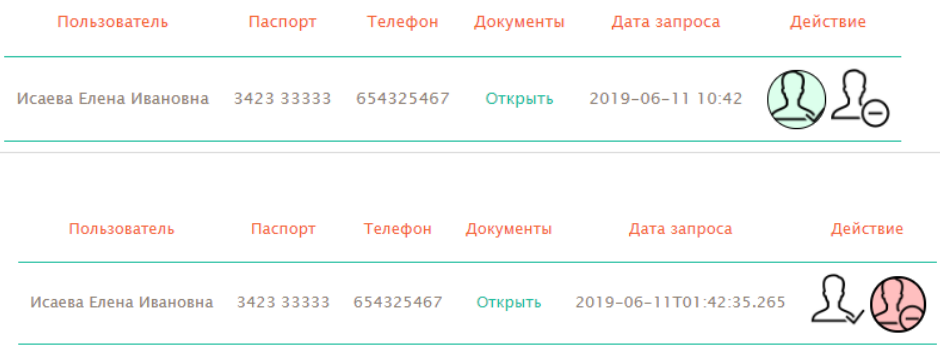

Рисунок 50 – Подтверждение/отклонение заявки

В случае отмены, выводится сообщение, подтверждая, заявка становится неактивной и исчезает из таблицы как показано на рисунке 51.

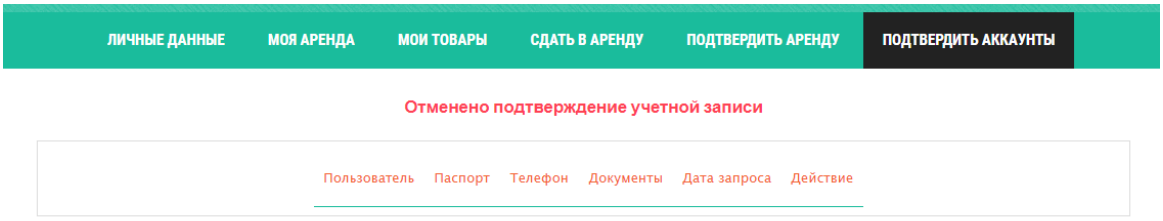

Рисунок 51 – Отклонение подтверждения

При этом у пользователя появится сообщение о неудачной попытки и возможно повторить отправку данных заново в соответствии с рисунком 52.

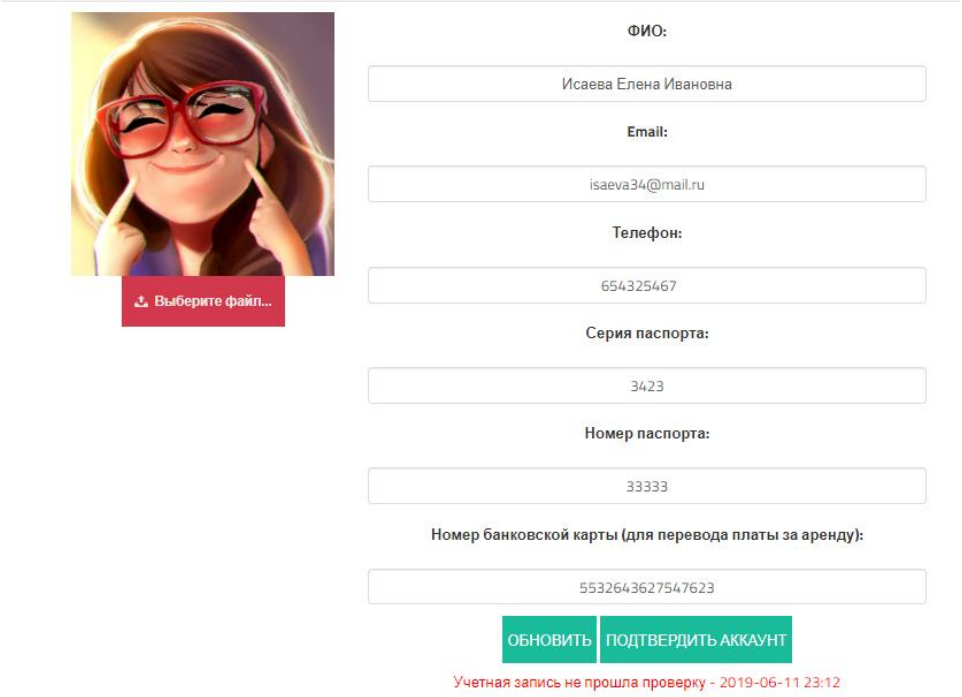

Рисунок 52 – Сообщение о не пройдённой проверки

Иначе, у пользователя отобразится одобрительное сообщение и пропадет кнопка для подтверждения аккаунта в соответствии с рисунком 53.

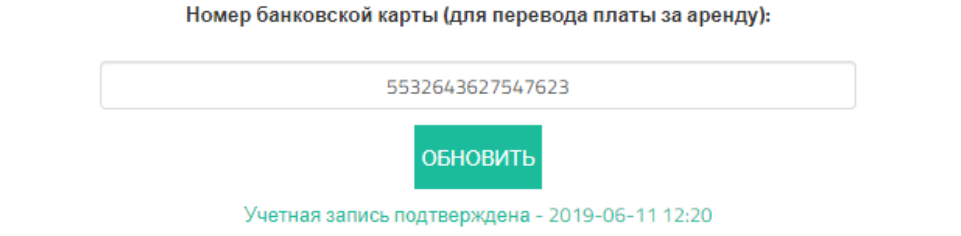

Рисунок 53 – Сообщение об успешном подтверждении

Теперь возьмем товар в аренду и у арендодателя появится новая заявка на аренду как показано на рисунке 54.

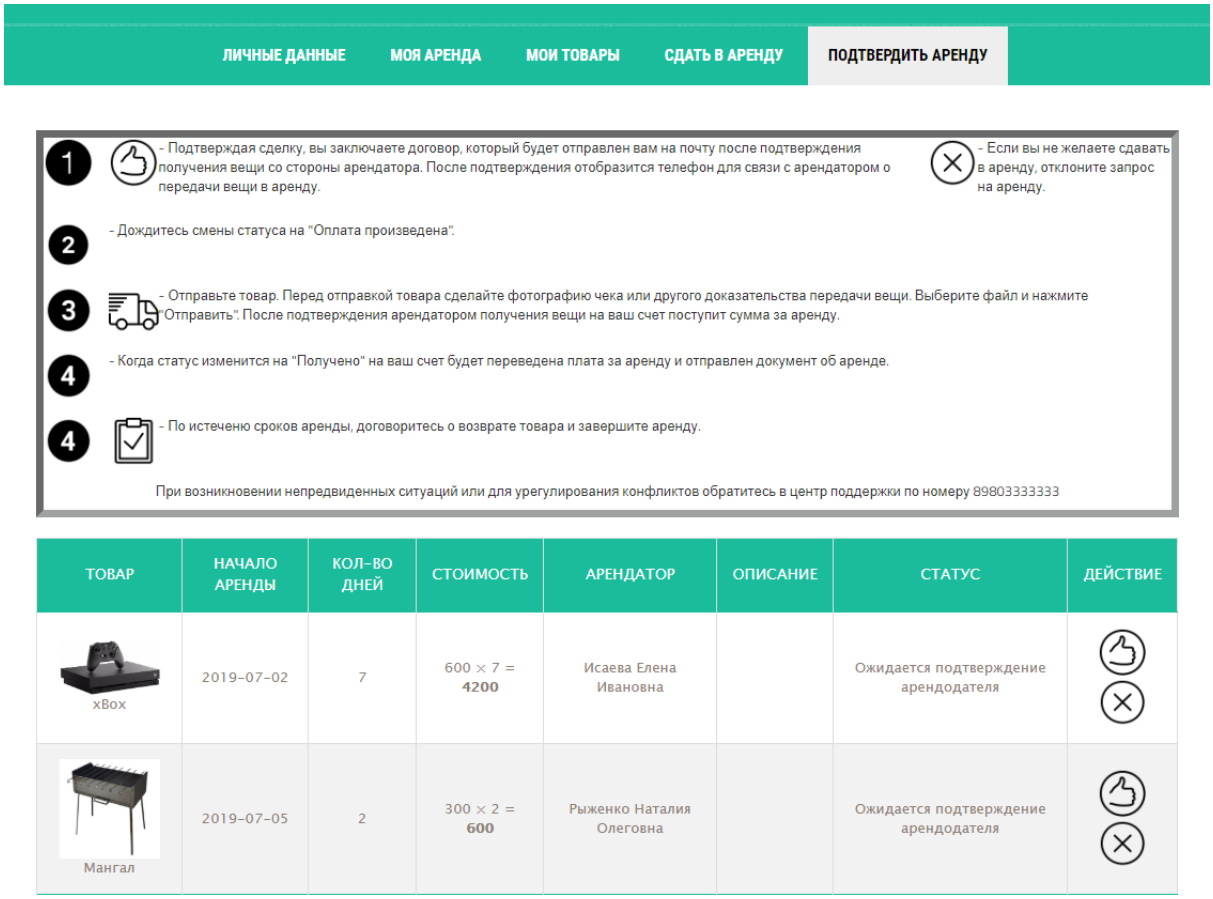

Рисунок 54 – Личный кабинет арендодателя

При подтверждении аренды запрашивается способ подписания договора. Окошко выбора способа подписания договора представлено на рисунке 55.

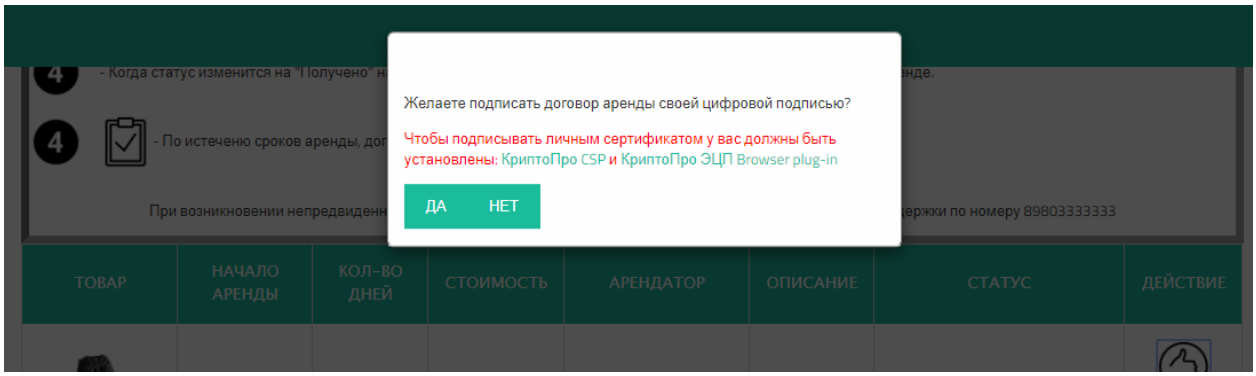

Рисунок 55 – Определение способа подписания договора

Если у пользователя нет квалифицированного сертификата, то в данном случае будет использовано подписание неквалифицированной цифровой

подписью, которую web-сервис сгенерирует для пользователя и сохранит. Для имеющих квалифицированный сертификат можно воспользоваться функционалом по подписании договора квалифицированной подписью. Чтобы проверить, что все требуемые ПО установлены и сертификат загружен в хранилище сертификатов на компьютере, нужно перейти по выделенной ссылке, которая представлена в диалоговом окне на рисунке 56.

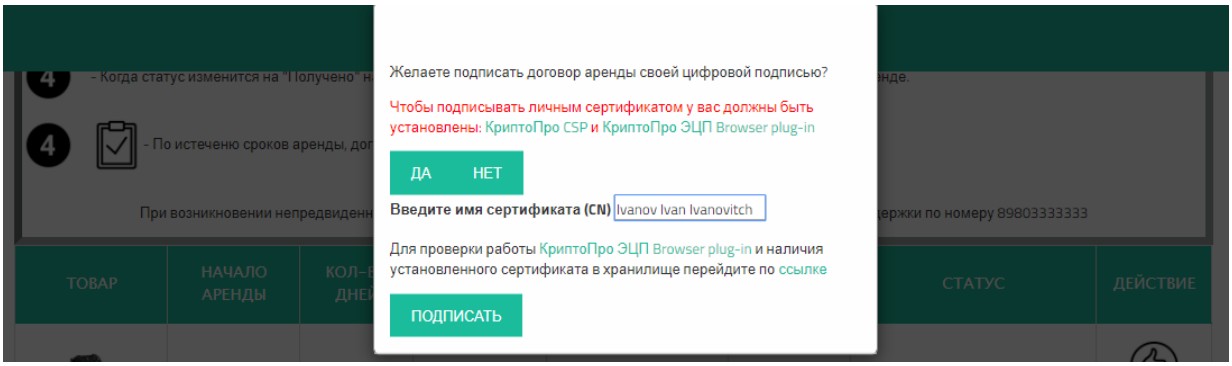

Рисунок 56 – Диалоговое окно с подсказками

Перейдя по ссылке, будет выведен список имеющихся сертификатов как показано на рисунке 57 и информация о них, с помощью которых можно подписать договор.

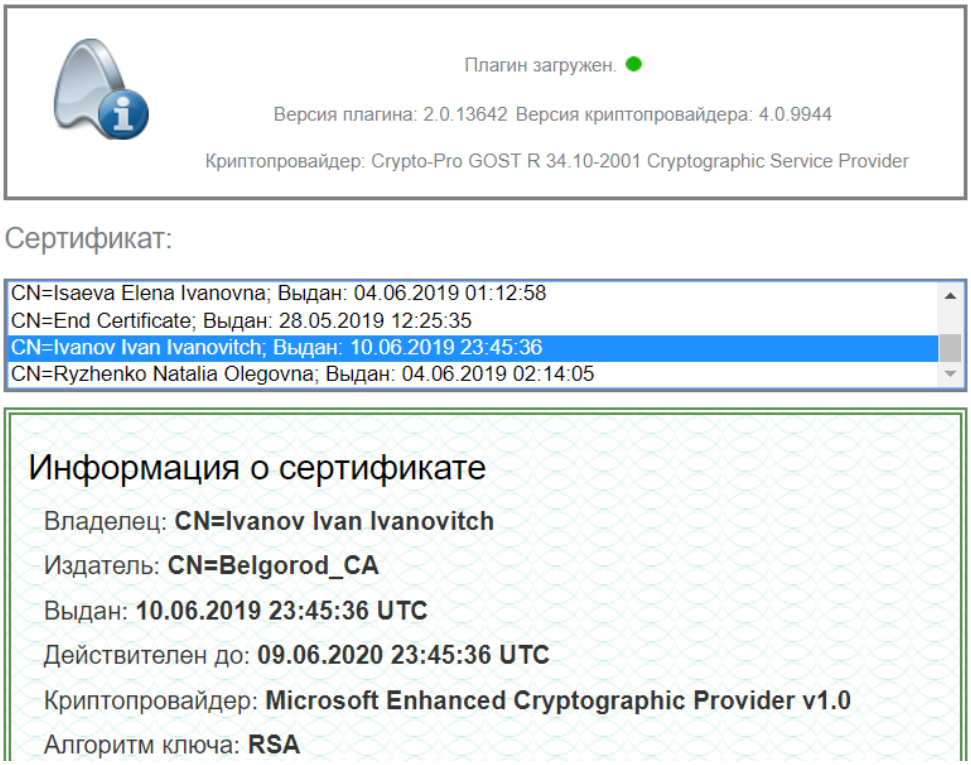

Рисунок 57 – Проверка работы плагина

После ввода имени сертификата (CN) и подтверждения подписи, произойдет подписание. Результат будет выведен на экран в соответствии с рисунком 58.

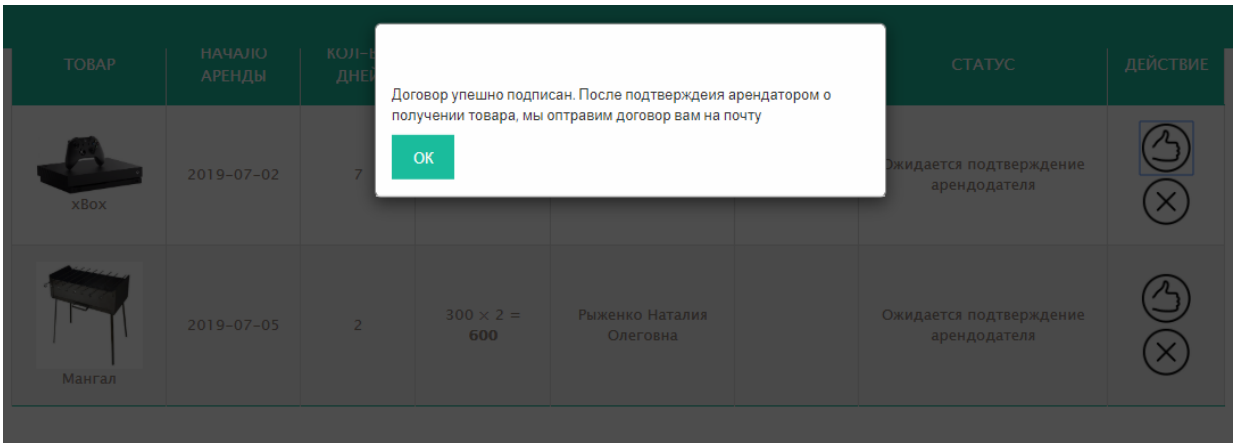

Рисунок 58 – Успешное подписание договора

После аналогичного подписания, арендатору нужно оплатить аренду на упомянутые в инструкции реквизиты и загрузить чек на оплату по нажатию на кнопку «Выберите файл», которая представлена на рисунке 59.

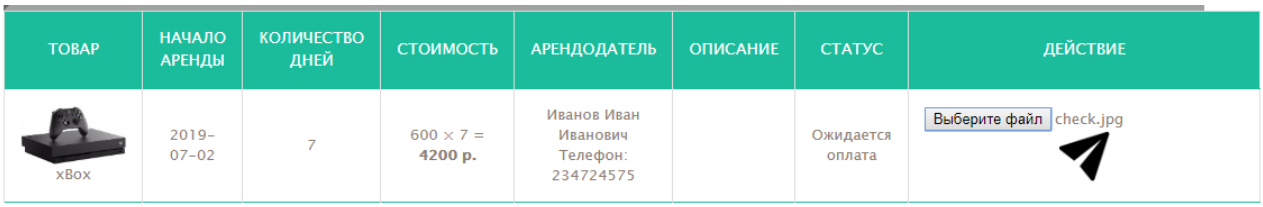

Рисунок 59 – Загрузка чека

Если же был загружен неверный формат, система выдаст сообщение как на рисунке 60.

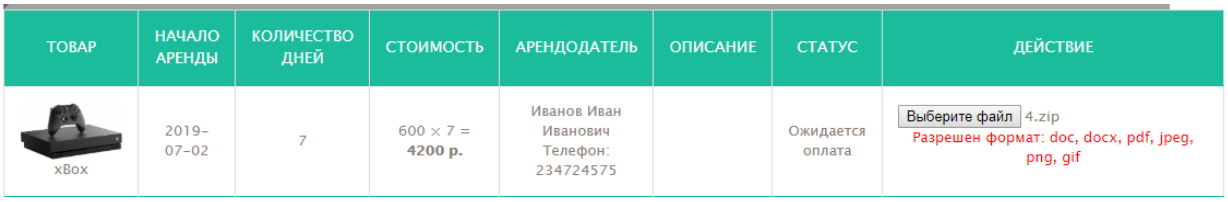

Рисунок 60 – Разрешенные форматы

Арендодателю остается отправить товар и загрузить отчетность об отправке как показано на рисунке 61.

Рисунок 61 – Запрос на отправку товара

В ходе загрузки документов, каждая сторона может скачать чек об оплате или отчетность об отправке, чтобы убедиться в правильности выполнения действий другой стороны. По нажатию на ссылки появится диалог для скачивания файлов как показано на рисунке 62.

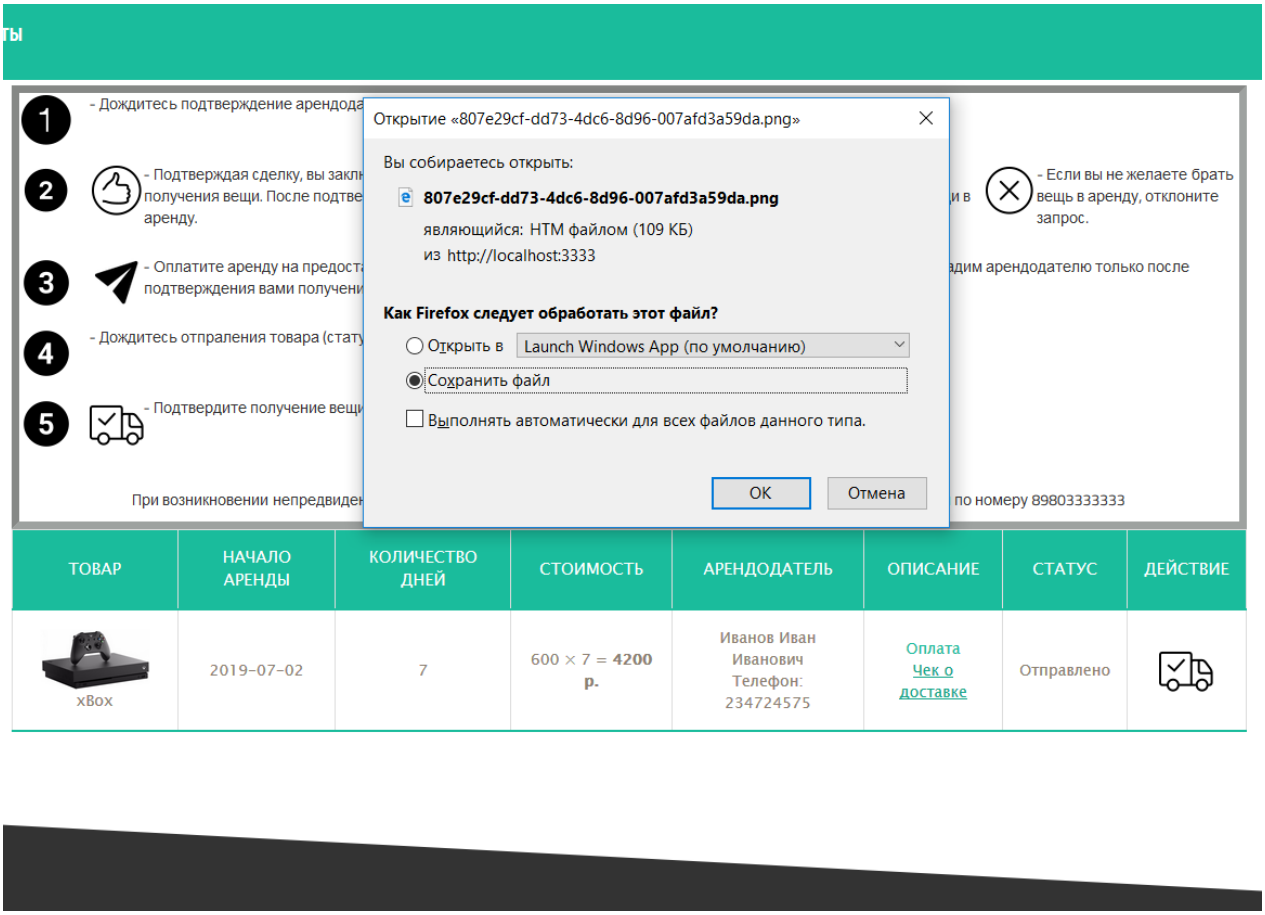

Рисунок 62 – Скачивание арендатором отчетности об отправке

После подтверждения получения товара, обе стороны получают договор об аренде, представленный на рисунке 63.

Договор аренды №1
движимого имущества

г.Белгород **2019-06-11**

Настоящий договор заключается посредством использования сервиса RentZone (далее - Сервис), расположенного в сети Интернет по адресу: www.rentzone.com (далее по тексту-Сайт), на условиях Соглашения https://rentzone.com/public_offer, принятого следующими Сторонами договора:

Владелец (арендодатель) (сторона, сдающая вещь в аренду):

Ф.И.О.: Иванов Иван Иванович
Моб.телефон 234724575
Логин для входа в личный кабинет www.rentzone.com: user1@mail.ru
Паспорт (серия, номер) ЕП 4534

Digitally signed by Ivanov Ivan Ivanovitch
Date: 2019.06.11 03:28:10 MSK
Reason: sign contracts
Location: Belgorod

с одной стороны,
и

Арендатор (сторона, берущая вещь в аренду):

Ф.И.О.: Исаева Елена Ивановна
Моб.телефон 654325467
Логин для входа в личный кабинет www.rentzone.com: isaeva34@mail.ru
Паспорт (серия, номер) 3423 33333

с другой стороны,

Digitally signed by Isaeva Elena Ivanovna
Date: 2019.06.11 03:29:38 MSK
Reason: sign contracts
Location: Belgorod

о нижеследующем:

1. ПРЕДМЕТ ДОГОВОРА

1.1. Владелец обязуется предоставить во временное владение и пользование движимое имущество (далее – Имущество):
Название: xBox

Рисунок 63 – Договор аренды

Договор заполняется основными данными пользователей и подписывается цифровой подписью, которые отображаются в виде текста с картинкой, по которому можно клацнуть и просмотреть с помощью какого сертификата был подписан договор. Просмотр сертификата пользователя представлен на рисунке 64.

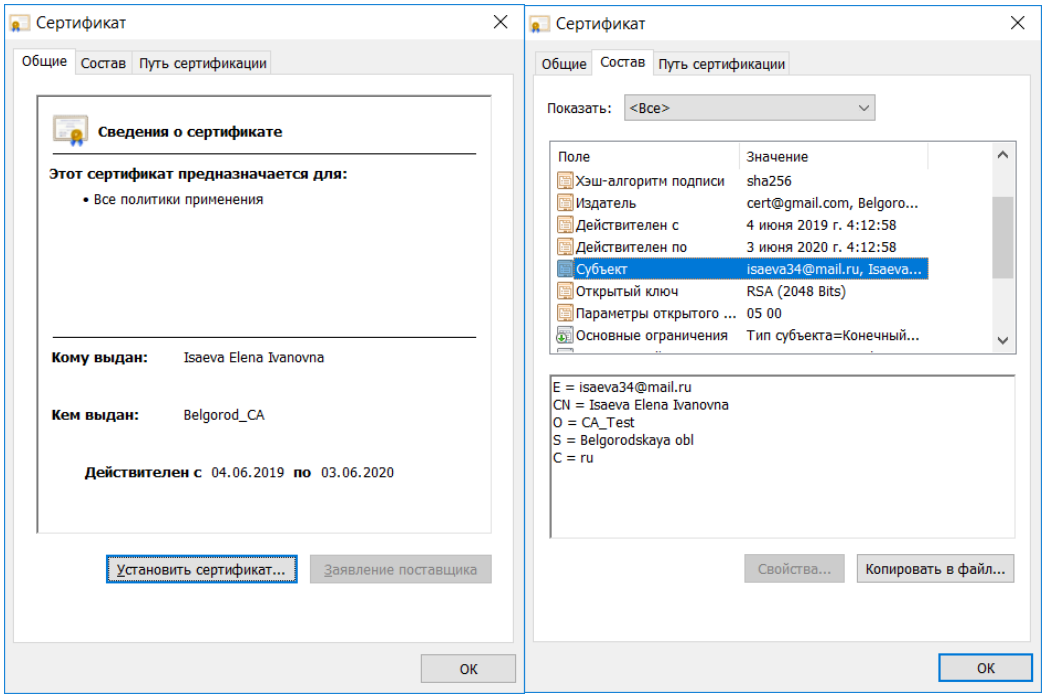

Рисунок 64 – Сертификат

Из сертификата можно идентифицировать личность, которая подписала договор и узнать не редактировался ли файл после подписания.

3.5 SWOT-анализ сильных и слабых сторон

SWOT-анализ представляет собой методику для обработки данных, целью которой является определение сильных и слабых сторон, угроз и возможностей объекта анализа. SWOT-матрица для web-сервиса по предоставлению товаров в аренду представлена в таблице 6.

Таблица 6 – SWOT-матрица

Сильные стороны	Возможности		Угрозы			Итого
	Совершенствование разработки	Привлечение новых клиентов	Увеличение количества конкурентов	Мошенничество	Спад спроса на аренду	
Использование квалифицированной подписи	+	++	0	+	0	4

Продолжение таблицы 6

Сильные стороны	Возможности		Угрозы			Итого
	Совершенствование разработки	Привлечение новых клиентов	Увеличение количества конкурентов	Мошенничество	Спад спроса на аренду	
Оформление online-договора аренды	+	++	0	+	+	5
Высокая скорость работы	+	+	0	0	0	2
Низкая вероятность ошибки	+	+	0	0	0	2
Широкий ассортимент товара	0	++	0	0	+	3
Приятный дизайн сайта	0	+	0	0	+	2
Итого	4	9	0	2	3	+18
Слабые стороны						
Рентабельность зависит от количества успешных сделок	-	0	0	0	0	1
Необходимость сопровождения системы	0	-	0	0	0	1
Итого	1	1	0	0	0	-2
Общий итог	+					+16

Таблица 6 демонстрирует, что достоинства web-сервиса преобладают над недостатками. Также из таблицы следует, что самыми важными достоинствами сервиса являются оформление online-договора и использование квалифицированной подписи.

Вывод по третьему разделу

В данном разделе была создана база данных, реализована серверная и клиентская часть, проведено тестирование web-сервиса и проведен SWOT-анализ.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был разработан web-сервис для предоставления товаров в аренду. В процессе разработки и проектирования были решены следующие задачи:

– исследовано заключение договора аренды по российскому гражданскому праву;

– рассмотрены способы создания и применения цифровой подписи;

– учтен федеральный закон "Об электронной подписи";

– изучено существующие сервисы по предоставлению услуг аренды;

– рассмотрены существующие вспомогательные программы/библиотеки для создания сертификатов и цифровых подписей;

– разработан web-сервис для предоставления услуги защищенного онлайн-оформления договора аренды.

В ходе первых тестирований программы были выявлены незначительные ошибки в проектировании и реализации, которые были устранены. Программа успешно работает в стандартном режиме эксплуатации, а также при различных несанкционированных действиях пользователя.

В результате была достигнута основная цель работы: было разработано средство, позволяющее быстро и безопасно заключать сделки по аренде товаров, в том числе за счет развитого функционала работы с личным кабинетом арендатора/арендодателя, применения средств электронной подписи при заключении и исполнении договоров, отслеживания статуса перевода денежных средств между участниками сделки.

В будущем, возможны улучшения разработанной системы в следующих направлениях:

– добавление google-карты для отметки местонахождения товара;

– локализация web-сервиса;

– конвертация валют.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Sharing economy или жизнь в аренду. На чем экономят и чем делятся жители мегаполиса [Электронный ресурс]: – Режим доступа: https://vc.ru/flood/60010-sharing-economy-ili-zhizn-v-arendu-na-chem-ekonomyat-i-chem-delyatsya-zhiteli-megapolisa.

2. Гардероб в аренду: сколько можно заработать на прокате одежды [Электронный ресурс]: – Режим доступа: https://www.rbc.ru/own_business/10/06/2016/575a75649a79473b46211de6.

3. Законы, кодексы и нормативно-правовые акты РФ [Электронный ресурс]: – Режим доступа: https://legalacts.ru/doc/FZ-ob-jelektronnoj-podpisi/.

4. Гонгало, Б.М. Гражданское право / Под ред. Б.М. Гонгало. - М.: Статут, 2017. – 511 с.

5. Фомичёв, В. М. Криптографические методы защиты информации в 2 ч. Часть 1. Математические аспекты: учебник для академического бакалавриата / В. М. Фомичёв, Д. А. Мельников; под редакцией В. М. Фомичёва. – М.: Юрайт, 2018. – 209 с.

6. Виды электронной подписи и способы ее использования [Электронный ресурс]: – Режим доступа: https://tensor.ru/uc/ep/vidyi.

7. Суркова, Н.Е. Методология структурного проектирования информационных систем: Монография / Н.Е. Суркова, А.В. Остроух. – Красноярск: Научно-инновационный центр, 2014. – 190 с.

8. Остроух, А.В. Проектирование информационных систем / А.В. Остроух, Н.Е. Суркова. – СПб.: Лань, 2019. –164 с.

9. Репин, В.В. Бизнес-процессы. Моделирование, внедрение, управление / В. В. Репин. – М.: Манн, Иванов и Фербер, 2014. – 512 с.

10. Жданов, С.А. Информационные системы: учебник для студ. учреждений высш. образования / С.А. Жданов, М.Л. Соболева, А.С. Алфимова. – М.: ООО «Прометей», 2015. – 302 с.

11. Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: учеб. пособие: Учебное пособие / В.Ю. Пирогов. – СПб: БХВ-Петербург, 2009. – 528 с.

12. Преснякова, Г.В. Проектирование интегрированных реляционных баз данных / Г.В. Преснякова. – М.; СПб.: КДУ; Петроглиф, 2011. – 131с.

13. Целигорова, Е.Н. Современные информационные технологии и их использование для исследования систем автоматического управления / Е.Н. Целигорова. – М.: Инженерный вестник Дона, 2010. – 144 с.

14. Кострова, А.В. Методы и модели информационного менеджмента. Учебное пособие / А.В. Кострова – М.: Финансы и статистика, 2010. – 336 с.

15. Монахов, В.В. Язык программирования Java и среда NetBeans / В.В Монахов. – СПб.: «БХВ-Петербург», 2010. – 640 с.

16. Давыдов, С. IntelliJ IDEA. Профессиональное програмирование на Java / С. Давыдов, А. Ефимов. – М: Вильямс, 2011. – 800 с.

17. Волков, А.С. Изучаем PostgreSQL 10 / А.С. Волков, Д. Салахалдин. – М.: ДМК Пресс, 2019. – 400 с.

18. Рыбанов, А.А. Разработка web-ориентированной информационной системы мониторинга и управления процессом прохождения производственной практики / А.А. Рыбанов, А.В. Рыльков. – М.: Молодой ученый, 2013. – 36 с.

19. Дронов, В. А. - HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В.А. Дронов. – СПБ.: БХВ-Петербург, 2011 – 414 с.

20. Браун, Э. Изучаем JavaScript. Руководство по созданию современных веб-сайтов / Э. Браун. – М.: Альфа-книга, 2017. –368 с.

21. Фримен, Э. Изучаем HTML, XHTML и CSS 2-е изд. / Э. Фримен. – СПб.: Питер, 2013. – 720 с.

22. Флэнаган, Д. JavaScript. Подробное руководство / Д. Флэнаган. – М.: Символ-Плюс, 2012. – 1080 с.

23. Хорстманн, К. Java. Библиотека профессионала. Том 1. Основы / К. Хорстманн. – М.: «Вильямс», 2018. – 864 с.

24. Уоллс, К. Spring в действии / К. Уоллс. – М.: ДМК Пресс, 2015. – 754 с.

25. Чугреев, В.Л. Особенности реализации MVC-архитектуры в веб-приложениях / В.Л. Чугреев. – М.: Молодой ученый, 2015. – 71 с.

26. Пьюривал, С. Основы разработки веб-приложений / С. Пьюривал. – СПб.: Питер, 2015. – 272 с.

27. Сьерра, К. Изучаем Java / К. Сьерра, Б. Бейтс. – М.: Эксмо, 2018. – 720 с.

28. Лафоре, Р. Структуры данных и алгоритмы в Java / Р. Лафоре. - СПб.: Питер, 2018.- 704 с.

29. Лабберс, П. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений/ П.Лабберс. – М.: «Вильямс», 2011. – 453 с.

30. Эккель, Б. Философия Java / Б. Эккель. – СПб.: Питер, 2019. - 1168 с.

31. Маккоу, А. Веб-приложения на JavaScript / А. Маккоу. – М.:Эксмо, 2012. – 285с.

32. Гради, Б. Объектноориентированый анализ и проектирование с примерами приложений / Б. Гради, А. Роберт и др. – М.:Вильямс, 2010. – 720 с.

ПРИЛОЖЕНИЕ А

Разработка информационной модели

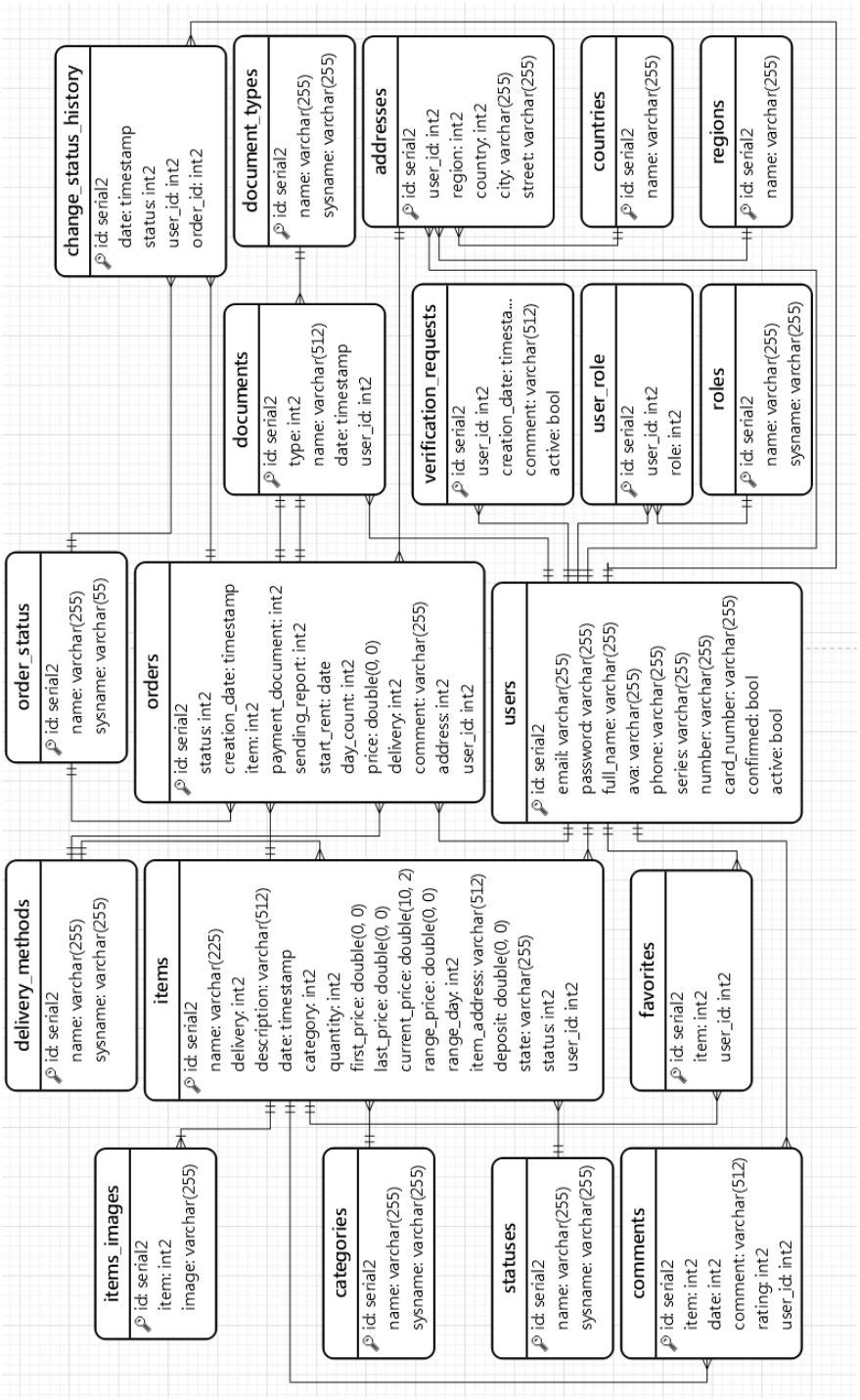

Рисунок А.1 – Физическая модель данных

ПРИЛОЖЕНИЕ В

Листинг кода по созданию web-сервиса

Листинг 1 – создание модели «Заявка на аренду товара»:

```
@Data
@Entity
@Table(name = "orders")
public class Order {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "user_id")
    private Integer user;

    private Integer product;

    private Integer quantity;

    @Convert(converter = TimeStampConverter.class)
    private LocalDateTime date;

    @Column(name = "start_rent")
    private LocalDate startRent;

    @JsonSerialize(using = BigDecimalSerializer.class)
    @JsonDeserialize(using = BigDecimalDeserializer.class)
    private BigDecimal price;

    @ManyToOne
    @JoinColumn(name = "address")
    private Address address;

    private String comment;

    private String recipient;

    private String phone;

    @Enumerated(EnumType.STRING)
    private Payment payment;

    @ManyToOne
    @JoinColumn(name="status")
    private OrderStatus status;

    @ManyToOne
```

```
    @JoinColumn(name = "sending_report")
    private Document sendingReport;

    @ManyToOne
    @JoinColumn(name = "payment_document")
    private Document paymentDocument;

    @ManyToOne
    @JoinColumn(name = "delivery")
    private DeliveryMethod delivery;

    public Order() {
    }

    public Order(Integer user, Integer product, Integer quantity, LocalDateTime date,
            BigDecimal price, Address address, String comment, String recipient,
            String phone, Payment payment, OrderStatus status, Document sendingReport,
            Document paymentDocument) {
        this.user = user;
        this.product = product;
        this.quantity = quantity;
        this.date = date;
        this.price = price;
        this.address = address;
        this.comment = comment;
        this.payment = payment;
        this.status = status;
        this.recipient = recipient;
        this.phone = phone;
        this.sendingReport = sendingReport;
        this.paymentDocument = paymentDocument;
    }
}
```

Листинг 2 – Создание OrderDao:

```
public interface OrderDAO extends CrudRepository<Order, Integer> {
    List<Order> findByUser(Integer userId);
    List<Order> findByProductInAndStatusNotIn(List<Integer> products, List<OrderStatus>
status);

    @Modifying
    @Query("update Order o set o.status = (Select s.id from OrderStatus s where s.sysname =
:status) where o.id = :id")
    void updateStatus(Integer id, String status);

    @Modifying
    @Query("update Order o set o.sendingReport.id = :sendingReport where o.id = :id")
    void updateSendingReport(Integer id, Integer sendingReport);
```

```
    @Modifying
    @Query("update Order o set o.paymentDocument.id = :document where o.id = :id")
    void updatePaymentDocument(Integer id, Integer document);
}
```

Листинг 3 – Создание OrderService:

```
@Service
public class OrderServiceImpl implements OrderService {
    private OrderDAO orderDAO;
    private OrderStatusDAO orderStatusDAO;
    private AddressService addressService;
    private ProductService productService;
    private DocumentService documentService;
    private UserDAO userDAO;
    private DocumentSignService documentSignService;
    @Value("${document.path}")
    private String basePathToDocument;

    @Autowired
    public OrderServiceImpl(OrderDAO orderDAO, ProductService productService,
AddressService addressService, OrderStatusDAO orderStatusDAO, UserDAO userDAO,
DocumentService documentService, DocumentSignService documentSignService) {

        this.orderDAO = orderDAO;
        this.productService = productService;
        this.addressService = addressService;
        this.orderStatusDAO = orderStatusDAO;
        this.userDAO = userDAO;
        this.documentService = documentService;
        this.documentSignService = documentSignService;
    }

    @Override
    @Transactional
    public void applyOrder(OrderFormDTO orderForm) {
```

```
        Integer userId = SecurityUtils.getUserId();
        orderForm.setUser(userId);
        OrderDTO orderDTO = new OrderDTO(orderForm, new OrderUserDTO(userId));
        orderDTO.setPrice(productService.read(orderDTO.getProduct().getId()).getCurrentPrice());
        orderDTO.setStatus(
            orderStatusDAO.findBySysname(
                OrderStatusValues.WAITING_LESSOR_APPROVEMENT
        ));

        orderDTO.setDate(LocalDateTime.now());

        if (Objects.nonNull(orderForm.getAddressId())) {
            orderDTO.setAddress(addressService.getAddressById(orderForm.getAddressId()));
        } else {
            orderDTO.setAddress(addressService.writeAddress(new AddressDTO(orderForm)));
        }
        Order order = new Order(orderDTO);
        orderDAO.save(order);
    }

    @Override
    public List<OrderDTO> getOrderHistory(Integer userId) {

        List<OrderDTO> orders = new ArrayList<>();

        orderDAO.findByUser(userId)
                .forEach(
                    order ->
                        orders.add(
                            new OrderDTO(
                                order,
                                new
OrderUserDTO(userDAO.findById(order.getUser()).orElseThrow(NotFoundException::new)),
                                productService.read(order.getProduct()),
                                new OrderUserDTO(userDAO.findByProductId(order.getProduct()))
                            )
                        )
                );
        return orders;
    }

    @Override
    public List<OrderDTO> fetchActiveOrders(Integer userId) {

        List<Integer> productsByUser = productService.getIdProductsByUser(userId);
        List<OrderDTO> orders = new ArrayList<>();

        List<Order> orderList = orderDAO.findByProductInAndStatusNotIn(
```

```
            productsByUser,
singletonList(orderStatusDAO.findBySysname((OrderStatusValues.DONE)))
        );

        orderList.forEach(
            order -> orders.add(
                new OrderDTO(
                    order,
                    new
OrderUserDTO(userDAO.findById(order.getUser()).orElseThrow(NotFoundException::new)),
                    productService.read(order.getProduct()), null
                )
            )
        );

        return orders;
    }

    @Transactional
    @Override
    public void changeStatus(Integer orderId, String action) {

        switch (action) {
            case "lessor_approve": orderDAO.updateStatus(orderId,
OrderStatusValues.WAITING_RENTER_APPROVEMENT);
            break;
            case "renter_approve": orderDAO.updateStatus(orderId,
OrderStatusValues.WAITING_PAYMENT);
            break;
            default: orderDAO.updateStatus(orderId, action);
        }

    }

    @Transactional
    @Override
    public void addDocument(Integer orderId, MultipartFile file, String documentType) {

        Document document = documentService.save(file, documentType);

        if (documentType.equals(DocumentTypeValues.SENDING_REPORT)) {
            orderDAO.updateSendingReport(orderId, document.getId());
            orderDAO.updateStatus(orderId, OrderStatusValues.ITEM_SENT);
        } else {
            orderDAO.updatePaymentDocument(orderId, document.getId());
            orderDAO.updateStatus(orderId, OrderStatusValues.PAID);
        }
    }

    @Override
    @Transactional
    public File createContract(final Integer userId, final Integer orderId, final String action) throws
```

```
Exception {
        changeStatus(orderId, action);
        User user = userDAO.findById(userId).orElseThrow(NotFoundException::new);
        if (!new File(basePathToDocument + "user_contract/temp_" + orderId +
"_signed.pdf").isFile()) {
            String fileName = createPdf(user, orderId, action);
            return documentSignService.sign(fileName, orderId, user, true);
        }
        String pathname = basePathToDocument + "user_contract/temp_" + orderId + "_signed";
        return documentSignService.sign(pathname, orderId, user, false);
    }

    @Override
    public String createContractWithExternal(Integer userId, Integer orderId, String action) throws
Exception {

        User user = userDAO.findById(userId).orElseThrow(NotFoundException::new);
        String fileName = createPdf(user, orderId, action);
        return documentSignService.signExternal(fileName + ".pdf", orderId);
    }

    @Override
    public File insertSign(final Integer userId, final Integer orderId, final String signedBytes) throws
Exception {
        String fileName = basePathToDocument + "user_contract/contract_" + orderId;
        return documentSignService.insertSign(fileName + ".pdf", orderId, signedBytes);
    }

    private String createPdf(User user, Integer orderId, String action) throws IOException {

        User renter;
        User owner;
        Order order = orderDAO.findById(orderId).orElseThrow(NotFoundException::new);
        if (action.equals("lessor_approve")) {
            owner = user;
            renter = userDAO.findByOrderId(orderId);
        } else {
            renter = user;
            owner = userDAO.findByProductId(order.getProduct());
        }
        final FileInputStream file = FileUtils.openInputStream(
            new File(basePathToDocument + "rent_contract.docx")
        );
        final XWPFDocument workbook = new XWPFDocument(file);
        List<XWPFParagraph> paragraphs = workbook.getParagraphs();
        //Пдстановка номера договора
        XWPFRun contractNumber = paragraphs.get(0).getRuns().get(0);
        contractNumber.setText(contractNumber.getText(0).replace("_____", "1"), 0);
        //Подстановка даты
        paragraphs.get(3).getRuns().get(9).setText(DateHelper.getCurrentDate(), 0);

        paragraphs.get(9).getRuns().get(2).setText(owner.getFullName(), 0);
```

```
        paragraphs.get(10).getRuns().get(1).setText(" " + owner.getPhone(), 0);
        paragraphs.get(11).getRuns().get(4).setText(owner.getEmail(), 0);
        paragraphs.get(12).getRuns().get(2).setText(owner.getSeries() + " " + owner.getNumber(), 0);
        paragraphs.get(19).getRuns().get(2).setText(renter.getFullName(), 0);
        paragraphs.get(20).getRuns().get(1).setText(" " + renter.getPhone(), 0);
        paragraphs.get(21).getRuns().get(4).setText(renter.getEmail(), 0);
        paragraphs.get(22).getRuns().get(2).setText(renter.getSeries() + " " + renter.getNumber(), 0);

        Product product = productService.findById(order.getProduct());

        paragraphs.get(31).getRuns().get(1).setText(product.getName(), 0);
        paragraphs.get(32).getRuns().get(1).setText(product.getDescription(), 0);
        paragraphs.get(42).getRuns().get(2).setText(order.getQuantity() + " суток с", 0);
        paragraphs.get(42).getRuns().get(3).setText(" " + order.getStartRent() + "г. по ", 0);
        paragraphs.get(42).getRuns().get(4).setText(new DateTime().toLocalDate().toString(), 0);
        paragraphs.get(51).getRuns().get(3).setText(order.getDelivery().getName() + " ", 0);
        paragraphs.get(58).getRuns().get(3).setText(order.getPrice().toString(), 0);
        paragraphs.get(59).getRuns().get(2).setText("обеспечительнный платеж", 0);

        file.close();

        String fileName = basePathToDocument + "user_contract/temp_" + orderId;
        final FileOutputStream outFile = new FileOutputStream(fileName + ".docx");

        workbook.write(outFile);
        outFile.close();

        try (InputStream is = new FileInputStream(new File(fileName + ".docx"));
             OutputStream out = new FileOutputStream(new File(fileName + ".pdf"))) {
            XWPFDocument document = new XWPFDocument(is);
            PdfOptions options = PdfOptions.getDefault();
            PdfConverter.getInstance().convert(document, out, options);
        } catch (Throwable e) {
            e.printStackTrace();
        }

        new File(fileName + ".docx").delete();
        return fileName;
    }
}
```

Листинг 4 – Создание контроллера:

```
@RestController
@RequestMapping("/api")
public class OrderController {

    private ProductService productService;
```

```
    private OrderService orderService;
    private AddressService addressService;

    @Autowired
    public OrderController(ProductService productService, OrderService orderService,
AddressService addressService) {
        this.productService = productService;
        this.orderService = orderService;
        this.addressService = addressService;
    }

    @GetMapping("/order/active")
    public List<OrderDTO> fetchActiveOrders() {
        return orderService.fetchActiveOrders(SecurityUtils.getUserId());
    }

    @PutMapping("/rent/{orderId}")
    public MessageDTO changeStatus(@PathVariable("orderId") Integer orderId, @RequestParam
String action) {
        orderService.changeStatus(orderId, action);
        return new MessageDTO("Статус успешно изменен");
    }

    @PostMapping
    @RequestMapping(value = "/rent/{orderId}/contract", produces =
APPLICATION_OCTET_STREAM_VALUE)
    public byte[] createContract(@PathVariable Integer orderId, @RequestParam String action)
throws Exception {
        return Files.toByteArray(orderService.createContract(SecurityUtils.getUserId(), orderId,
action));
    }

    @PutMapping
    @RequestMapping(value = "/rent/{orderId}/external-contract", produces =
APPLICATION_JSON_VALUE)
```

```
    public String createContractWithExternal(@PathVariable Integer orderId, @RequestParam
String action) throws Exception {
        return orderService.createContractWithExternal(SecurityUtils.getUserId(), orderId, action);
    }

    @PutMapping
    @RequestMapping(value = "/rent/{orderId}/insert-sign", produces =
APPLICATION_OCTET_STREAM_VALUE)
    public byte[] insertSign(@PathVariable Integer orderId, @RequestParam String action) throws
Exception {

        orderService.changeStatus(orderId, action);
        return Files.toByteArray(orderService.createContract(SecurityUtils.getUserId(), orderId,
action));
    }

    @PutMapping("/rent/{orderId}/payment-document")
    public MessageDTO addPaymentDocument(@PathVariable("orderId") Integer orderId,
                                         @RequestBody MultipartFile paymentDocument) {

        orderService.addDocument(orderId,
paymentDocument,DocumentTypeValues.PAYMENT_DOCUMENT);
        return new MessageDTO("Статус успешно изменен");
    }

    @PutMapping("/rent/{orderId}/sending-report")
    public MessageDTO addSendingReport(@PathVariable("orderId") Integer orderId,
                                       @RequestBody MultipartFile sendingReport) {

        orderService.addDocument(orderId, sendingReport,
DocumentTypeValues.SENDING_REPORT);
        return new MessageDTO("Статус успешно изменен");
    }

    @RequestMapping(value = "/{productId}/order")
```

```
    public ModelAndView checkoutPage(@PathVariable("productId") Integer productId) {

        ModelAndView model = new ModelAndView("checkout");
        ProductDTO product = productService.read(productId);
        List<AddressDTO> addresses = addressService.getAllAddresses(SecurityUtils.getUserId());
        model.addObject("product", product);
        model.addObject("addresses", addresses);
        model.addObject("pageOrder", UUID.randomUUID().toString());
        return model;
    }
    @RequestMapping(value = "/{productId}/rent", method = RequestMethod.POST, produces =
APPLICATION_JSON_VALUE)
    public ResponseEntity<MessageDTO> rentItem(@PathVariable Integer productId,
                                    @RequestBody @Validated OrderFormDTO orderForm,
HttpSession session) {

        orderForm.setProductId(productId);
        if (!isNewOrder(orderForm.getOrderPageId(), session)) {
            return new ResponseEntity<>(new MessageDTO("You already accept this order!"),
HttpStatus.ALREADY_REPORTED);
        }
        session.setAttribute("lastOrderPageId", orderForm.getOrderPageId());
        orderService.applyOrder(orderForm);
        return new ResponseEntity<>(new MessageDTO("Thanks, your order has been sent for
processing!"), HttpStatus.OK);
    }

    private boolean isNewOrder(String orderPageId, HttpSession session) {

        String lastOrderPageId
            = (String) Optional.ofNullable(session.getAttribute("lastOrderPageId")).orElse("");
        return !orderPageId.equals(lastOrderPageId);
    }
}
```

Листинг 5 – Сервис по созданию сертификатов и цифровых подписей:

```
package com.epam.popshop.service.impl;
import com.epam.popshop.models.User;
import com.epam.popshop.service.DocumentSignService;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.Image;
import com.itextpdf.text.Rectangle;
import com.itextpdf.text.pdf.*;
import com.itextpdf.text.pdf.security.*;
import org.apache.commons.codec.binary.Base64;
import org.bouncycastle.asn1.pkcs.PrivateKeyInfo;
import org.bouncycastle.cert.X509CertificateHolder;
import org.bouncycastle.cert.jcajce.JcaX509CertificateConverter;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.openssl.PEMParser;
import org.bouncycastle.openssl.jcajce.JcaPEMKeyConverter;
import org.bouncycastle.openssl.jcajce.JceOpenSSLPKCS8DecryptorProviderBuilder;
import org.bouncycastle.operator.InputDecryptorProvider;
import org.bouncycastle.operator.OperatorCreationException;
import org.bouncycastle.pkcs.PKCS8EncryptedPrivateKeyInfo;
import org.bouncycastle.pkcs.PKCSException;
import org.bouncycastle.pkcs.bc.BcPKCS12PBEInputDecryptorProviderBuilder;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.io.*;
import java.security.GeneralSecurityException;
import java.security.PrivateKey;
import java.security.Security;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

@Service
public class DocumentSignServiceImpl implements DocumentSignService {

   @Value("${document.path}")
   private String basePathToDocument;

   private java.security.cert.X509Certificate readCACert(Integer userId) throws IOException,
CertificateException {
      final File caCertFile = new File("documents/certificate/" + userId + "_cert.crt");

      try (FileReader caCertFileReader = new FileReader(caCertFile);
           PEMParser caCertReader = new PEMParser(caCertFileReader)) {

         X509CertificateHolder x509Certificate = (X509CertificateHolder)
caCertReader.readObject();

         return new JcaX509CertificateConverter().setProvider("BC" )
```

```
            .getCertificate( x509Certificate );
        }
    }

    private PrivateKey readCAPrivateKey(Integer userId) throws IOException, PKCSException,
OperatorCreationException {
        final File caPrivateKeyFile = new File("documents/certificate/" + userId + "_pk.pem");

        try (FileReader caPrivateKeyFileReader = new FileReader(caPrivateKeyFile);
             PEMParser pemParser = new PEMParser(caPrivateKeyFileReader)) {

            final PKCS8EncryptedPrivateKeyInfo
                keypair = (PKCS8EncryptedPrivateKeyInfo) pemParser.readObject();

            JceOpenSSLPKCS8DecryptorProviderBuilder jce = new
JceOpenSSLPKCS8DecryptorProviderBuilder();
            jce.setProvider("BC");
            InputDecryptorProvider decProv = jce.build("cthnbabrfn".toCharArray());
            PrivateKeyInfo info = keypair.decryptPrivateKeyInfo(decProv);

            return new JcaPEMKeyConverter().getPrivateKey(info);
        }
    }

    @Override
    @Transactional
    public File sign(String path, Integer orderId, User user, Boolean withLessorSign)
        throws IOException, GeneralSecurityException, DocumentException, PKCSException,
OperatorCreationException {

        Security.addProvider(new BouncyCastleProvider());

        PrivateKey key = readCAPrivateKey(user.getId());
        X509Certificate cert = readCACert(user.getId());
        MakeSignature.CryptoStandard subfilter = MakeSignature.CryptoStandard.CMS;

        PdfReader reader = new PdfReader(path + ".pdf");
        String newPath = path + "_signed.pdf";
        FileOutputStream os = new FileOutputStream(new File(newPath));
        PdfStamper stamper = PdfStamper.createSignature(reader, os, '\0');

        Certificate[] certificates = new Certificate[] { cert};

        PdfSignatureAppearance appearance = stamper.getSignatureAppearance();

        try (FileInputStream fis = new FileInputStream(new File("images/cert2.png"));
             ByteArrayOutputStream baos = new ByteArrayOutputStream()){
            byte[] buf = new byte[1024];
            int n = 0;
            while ((n = fis.read(buf, 0, buf.length)) != -1) {
                baos.write(buf, 0, n);
                Image image = Image.getInstance(baos.toByteArray());
```

```
            appearance.setImage(image);

        }
    }


appearance.setCertificationLevel(PdfSignatureAppearance.CERTIFIED_NO_CHANGES_ALLOWED);
    appearance.setReason("sign contracts");
    appearance.setLocation("Belgorod");
    appearance.setContact(user.getPhone());
    appearance.setCertificate(cert);

    Rectangle signPlace = withLessorSign
        ? new Rectangle(406, 558, 554, 520)
        : new Rectangle(526, 358, 404, 400);

    appearance.setVisibleSignature(signPlace, 1, "sig" + user.getId());
    ExternalSignature pks = new PrivateKeySignature(key, DigestAlgorithms.SHA256, "BC");
    ExternalDigest digest = new BouncyCastleDigest();

    MakeSignature.signDetached(
        appearance, digest, pks, certificates, null, null, null, 0, subfilter
    );

    reader.close();
    stamper.close();
    os.close();
    new File(path + ".pdf").delete();
    return new File(newPath);
  }

  @Override
  @Transactional
  public String signExternal(String path, Integer orderId)
      throws IOException, DocumentException, GeneralSecurityException {

    Security.addProvider(new BouncyCastleProvider());
    PdfReader reader = new PdfReader(path);
    FileOutputStream os = new FileOutputStream(
        new File("documents/user_contract/cert/contract_" + orderId + ".pdf")
    );

    PdfStamper stamper = PdfStamper.createSignature(reader, os, '\0');
    PdfSignatureAppearance appearance = stamper.getSignatureAppearance();
    appearance.setVisibleSignature(new Rectangle(406, 558, 554, 520), 1, "sig");

    ExternalBlankSignatureContainer external =
        new ExternalBlankSignatureContainer(PdfName.ADOBE_PPKLITE,
PdfName.ADBE_PKCS7_DETACHED);

    MakeSignature.signExternalContainer(appearance, external, 65536);
```

```
        InputStream contentStream = appearance.getRangeStream();
        byte[] hash = DigestAlgorithms.digest(contentStream, DigestAlgorithms.SHA256, "BC");
        String hashString = bytesToHex(hash);
        reader.close();
        os.close();
        stamper.close();
        contentStream.close();
        return hashString;
    }

    private static String bytesToHex(byte[] hash) {
        StringBuilder hexString = new StringBuilder();
        for (byte hash1 : hash) {
            String hex = Integer.toHexString(0xff & hash1);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }
        return hexString.toString();
    }


    @Override
    @Transactional
    public File insertSign(String path, Integer orderId, String signedBytes)
        throws IOException, GeneralSecurityException, DocumentException {
        PdfReader reader = new PdfReader(path, null, true);
        byte[] decodedBytes = Base64.decodeBase64(signedBytes);
        ExternalSignatureContainer external = new MyExternalSignatureContainer(decodedBytes);
        String name = "documents/user_contract/cert/own_sign_contract_" + orderId + ".pdf";
        OutputStream stream = new FileOutputStream(name);
        MakeSignature.signDeferred(reader, "sig", stream, external);
        reader.close();
        stream.close();
        return new File(name);
    }
}

class MyExternalSignatureContainer implements ExternalSignatureContainer {
    private byte[] _signedBytes;
    public MyExternalSignatureContainer(byte[] signedBytes) {
        this._signedBytes = signedBytes;
    }

    @Override
    public byte[] sign(InputStream data) {
        return _signedBytes;
    }

    @Override
    public void modifySigningDictionary(PdfDictionary signDic) {
    }
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____